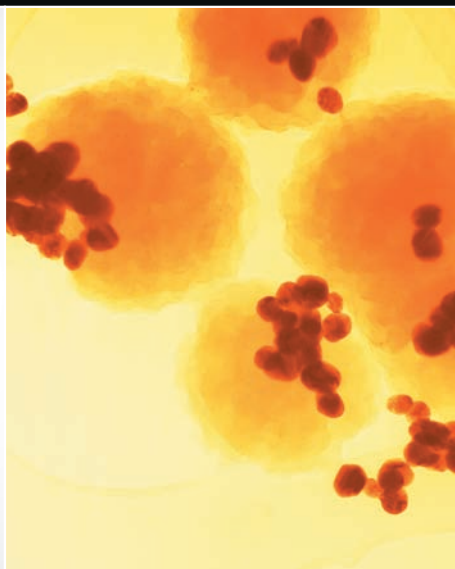




# Digital Image Processing

FOURTH EDITION

Rafael C. Gonzalez • Richard E. Woods

# Support Package for *Digital Image Processing*

Your new textbook provides access to support packages that may include reviews in areas like probability and vectors, tutorials on topics relevant to the material in the book, an image database, and more. Refer to the Preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for Rafael C. Gonzalez and Richard E. Woods' *Digital Image Processing*, Fourth Edition, Global Edition.

1. Go to **www.ImageProcessingPlace.com**
2. Find the title of your textbook.
3. Click Support Materials and follow the on-screen instructions to create a login name and password.

Use the login name and password you created during registration to start using the digital resources that accompany your textbook.

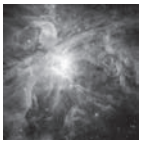
## IMPORTANT:

This serial code can only be used once. This subscription is not transferrable.

# Digital Image Processing





FOURTH EDITION

*Global Edition*


*Rafael C. Gonzalez*
University of Tennessee

*Richard E. Woods*
Interapptics



Pearson
330 Hudson Street, New York, NY 10013

*Senior Vice President Courseware Portfolio Management:* Marcia J. Horton
*Director, Portfolio Management: Engineering, Computer Science & Global Editions:* Julian Partridge
*Portfolio Manager:* Julie Bai
*Field Marketing Manager:* Demetrius Hall
*Product Marketing Manager:* Yvonne Vannatta
*Marketing Assistant:* Jon Bryant
*Content Managing Producer, ECS and Math:* Scott Disanno
*Content Producer:* Michelle Bayman
*Project Manager:* Rose Kernan
*Assistant Project Editor, Global Editions:* Vikash Tiwari
*Operations Specialist:* Maura Zaldivar-Garcia
*Manager, Rights and Permissions:* Ben Ferrini
*Senior Manufacturing Controller, Global Editions:* Trudy Kimber
*Media Production Manager, Global Editions:* Vikram Kumar
*Cover Designer:* Lumina Datamatics
*Cover Photo:* **CT image**—© zhuravliki.123rf.com/Pearson Asset Library; **Gram-negative bacteria**—© royaltystockphoto.com/Shutterstock.com; **Orion Nebula**—© creativemarc/Shutterstock.com; **Fingerprints**—© Larysa Ray/Shutterstock.com; **Cancer cells**—© Greenshoots Communications/Alamy Stock Photo

MATLAB is a registered trademark of The MathWorks, Inc., 1 Apple Hill Drive, Natick, MA 01760-2098.

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com



British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-22304-9
ISBN 13: 978-1-292-22304-9

Typeset by Richard E. Woods

Printed and bound in Malaysia

*To Connie, Ralph, and Rob*
*and*
*To Janice, David, and Jonathan*

*This page intentionally left blank*

# Contents

# 2 Digital Image Fundamentals

> Those who wish to succeed must ask the right preliminary questions.
>
> *Aristotle*

## *Preview*

This chapter is an introduction to a number of basic concepts in digital image processing that are used throughout the book. Section 2.1 summarizes some important aspects of the human visual system, including image formation in the eye and its capabilities for brightness adaptation and discrimination. Section 2.2 discusses light, other components of the electromagnetic spectrum, and their imaging characteristics. Section 2.3 discusses imaging sensors and how they are used to generate digital images. Section 2.4 introduces the concepts of uniform image sampling and intensity quantization. Additional topics discussed in that section include digital image representation, the effects of varying the number of samples and intensity levels in an image, the concepts of spatial and intensity resolution, and the principles of image interpolation. Section 2.5 deals with a variety of basic relationships between pixels. Finally, Section 2.6 is an introduction to the principal mathematical tools we use throughout the book. A second objective of that section is to help you begin developing a "feel" for how these tools are used in a variety of basic image processing tasks.

## *Upon completion of this chapter, readers should:*

- Have an understanding of some important functions and limitations of human vision.
- Be familiar with the electromagnetic energy spectrum, including basic properties of light.
- Know how digital images are generated and represented.
- Understand the basics of image sampling and quantization.
- Be familiar with spatial and intensity resolution and their effects on image appearance.
- Have an understanding of basic geometric relationships between image pixels.
- Be familiar with the principal mathematical tools used in digital image processing.
- Be able to apply a variety of introductory digital image processing techniques.

## 2.1 ELEMENTS OF VISUAL PERCEPTION

Although the field of digital image processing is built on a foundation of mathematics, human intuition and analysis often play a role in the choice of one technique versus another, and this choice often is made based on subjective, visual judgments. Thus, developing an understanding of basic characteristics of human visual perception as a first step in our journey through this book is appropriate. In particular, our interest is in the elementary mechanics of how images are formed and perceived by humans. We are interested in learning the physical limitations of human vision in terms of factors that also are used in our work with digital images. Factors such as how human and electronic imaging devices compare in terms of resolution and ability to adapt to changes in illumination are not only interesting, they are also important from a practical point of view.

### STRUCTURE OF THE HUMAN EYE

Figure 2.1 shows a simplified cross section of the human eye. The eye is nearly a sphere (with a diameter of about 20 mm) enclosed by three membranes: the *cornea* and *sclera* outer cover; the *choroid*; and the *retina*. The cornea is a tough, transparent tissue that covers the anterior surface of the eye. Continuous with the cornea, the sclera is an opaque membrane that encloses the remainder of the optic globe.

The choroid lies directly below the sclera. This membrane contains a network of blood vessels that serve as the major source of nutrition to the eye. Even superficial

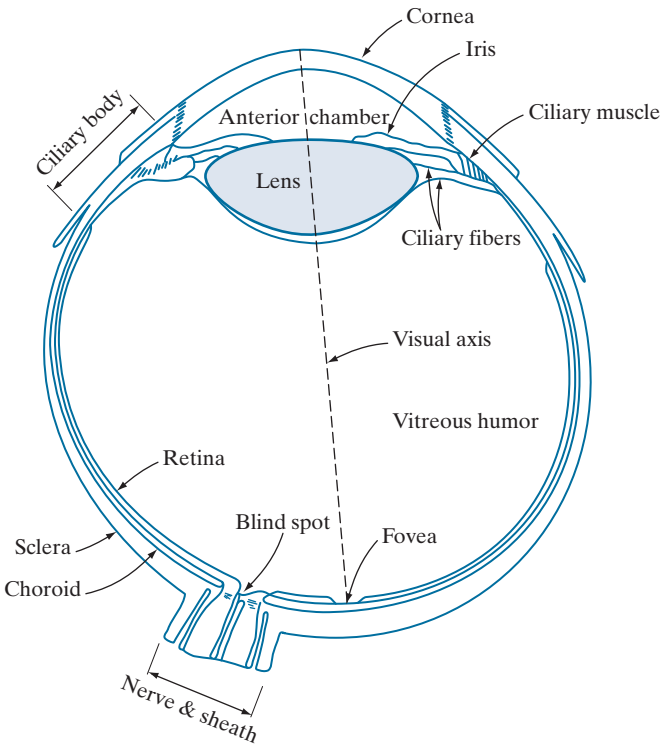



**FIGURE 2.1** Simplified diagram of a cross section of the human eye.

injury to the choroid can lead to severe eye damage as a result of inflammation that restricts blood flow. The choroid coat is heavily pigmented, which helps reduce the amount of extraneous light entering the eye and the backscatter within the optic globe. At its anterior extreme, the choroid is divided into the *ciliary body* and the *iris*. The latter contracts or expands to control the amount of light that enters the eye. The central opening of the iris (the *pupil*) varies in diameter from approximately 2 to 8 mm. The front of the iris contains the visible pigment of the eye, whereas the back contains a black pigment.

The *lens* consists of concentric layers of fibrous cells and is suspended by fibers that attach to the ciliary body. It is composed of 60% to 70% water, about 6% fat, and more protein than any other tissue in the eye. The lens is colored by a slightly yellow pigmentation that increases with age. In extreme cases, excessive clouding of the lens, referred to as *cataracts*, can lead to poor color discrimination and loss of clear vision. The lens absorbs approximately 8% of the visible light spectrum, with higher absorption at shorter wavelengths. Both infrared and ultraviolet light are absorbed by proteins within the lens and, in excessive amounts, can damage the eye.

The innermost membrane of the eye is the *retina*, which lines the inside of the wall's entire posterior portion. When the eye is focused, light from an object is imaged on the retina. Pattern vision is afforded by discrete light receptors distributed over the surface of the retina. There are two types of receptors: *cones* and *rods*. There are between 6 and 7 million cones in each eye. They are located primarily in the central portion of the retina, called the *fovea*, and are highly sensitive to color. Humans can resolve fine details because each cone is connected to its own nerve end. Muscles rotate the eye until the image of a region of interest falls on the fovea. Cone vision is called *photopic* or *bright-light* vision.

The number of rods is much larger: Some 75 to 150 million are distributed over the retina. The larger area of distribution, and the fact that several rods are connected to a single nerve ending, reduces the amount of detail discernible by these receptors. Rods capture an overall image of the field of view. They are not involved in color vision, and are sensitive to low levels of illumination. For example, objects that appear brightly colored in daylight appear as colorless forms in moonlight because only the rods are stimulated. This phenomenon is known as *scotopic* or *dim-light* vision.

Figure 2.2 shows the density of rods and cones for a cross section of the right eye, passing through the region where the optic nerve emerges from the eye. The absence of receptors in this area causes the so-called *blind spot* (see Fig. 2.1). Except for this region, the distribution of receptors is radially symmetric about the fovea. Receptor density is measured in degrees from the visual axis. Note in Fig. 2.2 that cones are most dense in the center area of the fovea, and that rods increase in density from the center out to approximately 20° off axis. Then, their density decreases out to the periphery of the retina.

The fovea itself is a circular indentation in the retina of about 1.5 mm in diameter, so it has an area of approximately 1.77 $mm^2$. As Fig. 2.2 shows, the density of cones in that area of the retina is on the order of 150,000 elements per $mm^2$. Based on these figures, the number of cones in the fovea, which is the region of highest acuity

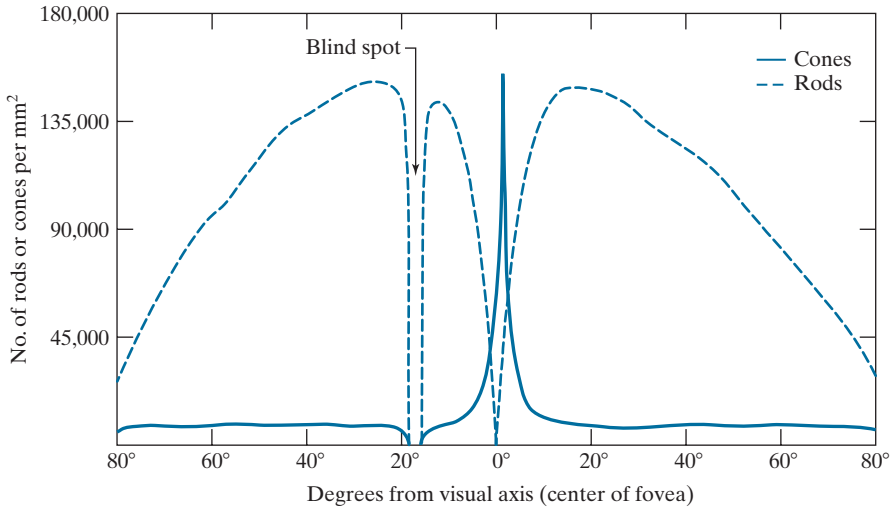



**FIGURE 2.2** Distribution of rods and cones in the retina.

in the eye, is about 265,000 elements. Modern electronic imaging chips exceed this number by a large factor. While the ability of humans to integrate intelligence and experience with vision makes purely quantitative comparisons somewhat superficial, keep in mind for future discussions that electronic imaging sensors can easily exceed the capability of the eye in resolving image detail.

## IMAGE FORMATION IN THE EYE

In an ordinary photographic camera, the lens has a fixed focal length. Focusing at various distances is achieved by varying the distance between the lens and the imaging plane, where the film (or imaging chip in the case of a digital camera) is located. In the human eye, the converse is true; the distance between the center of the lens and the imaging sensor (the retina) is fixed, and the focal length needed to achieve proper focus is obtained by varying the shape of the lens. The fibers in the ciliary body accomplish this by flattening or thickening the lens for distant or near objects, respectively. The distance between the center of the lens and the retina along the visual axis is approximately 17 mm. The range of focal lengths is approximately 14 mm to 17 mm, the latter taking place when the eye is relaxed and focused at distances greater than about 3 m. The geometry in Fig. 2.3 illustrates how to obtain the dimensions of an image formed on the retina. For example, suppose that a person is looking at a tree 15 m high at a distance of 100 m. Letting $h$ denote the height of that object in the retinal image, the geometry of Fig. 2.3 yields $15/100 = h/17$ or $h = 2.5$ mm. As indicated earlier in this section, the retinal image is focused primarily on the region of the fovea. Perception then takes place by the relative excitation of light receptors, which transform radiant energy into electrical impulses that ultimately are decoded by the brain.

## BRIGHTNESS ADAPTATION AND DISCRIMINATION

Because digital images are displayed as sets of discrete intensities, the eye's ability to discriminate between different intensity levels is an important consideration

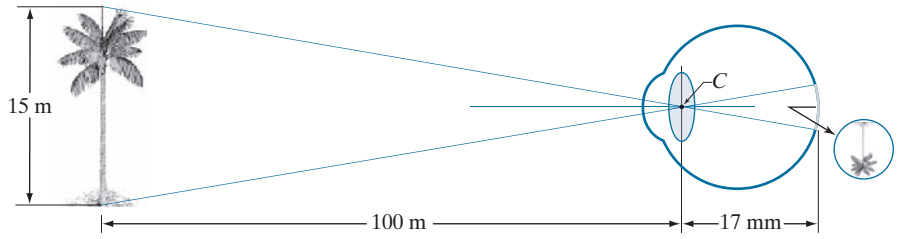



**FIGURE 2.3**
Graphical representation of the eye looking at a palm tree. Point $C$ is the focal center of the lens.

in presenting image processing results. The range of light intensity levels to which the human visual system can adapt is enormous—on the order of $10^{10}$— from the scotopic threshold to the glare limit. Experimental evidence indicates that *subjective brightness* (intensity as perceived by the human visual system) is a logarithmic function of the light intensity incident on the eye. Figure 2.4, a plot of light intensity versus subjective brightness, illustrates this characteristic. The long solid curve represents the range of intensities to which the visual system can adapt. In photopic vision alone, the range is about $10^6$. The transition from scotopic to photopic vision is gradual over the approximate range from 0.001 to 0.1 millilambert (–3 to –1 mL in the log scale), as the double branches of the adaptation curve in this range show.

The key point in interpreting the impressive dynamic range depicted in Fig. 2.4 is that the visual system cannot operate over such a range *simultaneously*. Rather, it accomplishes this large variation by changing its overall sensitivity, a phenomenon known as *brightness adaptation*. The total range of distinct intensity levels the eye can discriminate simultaneously is rather small when compared with the total adaptation range. For a given set of conditions, the current sensitivity level of the visual system is called the *brightness adaptation level*, which may correspond, for example,

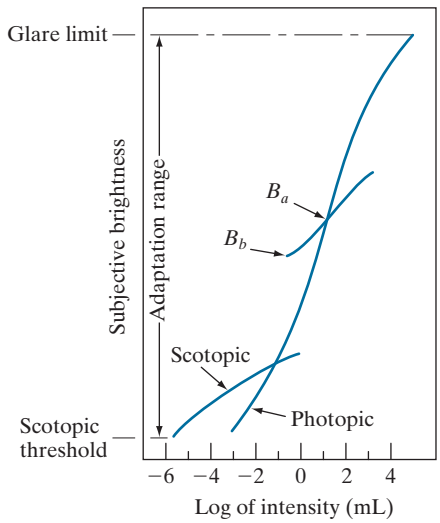



**FIGURE 2.4**
Range of subjective brightness sensations showing a particular adaptation level, $B_a$.

to brightness $B_a$ in Fig. 2.4. The short intersecting curve represents the range of subjective brightness that the eye can perceive when adapted to *this* level. This range is rather restricted, having a level $B_b$ at, and below which, all stimuli are perceived as indistinguishable blacks. The upper portion of the curve is not actually restricted but, if extended too far, loses its meaning because much higher intensities would simply raise the adaptation level higher than $B_a$.

The ability of the eye to discriminate between *changes* in light intensity at any specific adaptation level is of considerable interest. A classic experiment used to determine the capability of the human visual system for brightness discrimination consists of having a subject look at a flat, uniformly illuminated area large enough to occupy the entire field of view. This area typically is a diffuser, such as opaque glass, illuminated from behind by a light source, $I$, with variable intensity. To this field is added an increment of illumination, $\Delta I$, in the form of a short-duration flash that appears as a circle in the center of the uniformly illuminated field, as Fig. 2.5 shows.

If $\Delta I$ is not bright enough, the subject says "no," indicating no perceivable change. As $\Delta I$ gets stronger, the subject may give a positive response of "yes," indicating a perceived change. Finally, when $\Delta I$ is strong enough, the subject will give a response of "yes" all the time. The quantity $\Delta I_c / I$, where $\Delta I_c$ is the increment of illumination discriminable 50% of the time with background illumination $I$, is called the *Weber ratio*. A small value of $\Delta I_c / I$ means that a small percentage change in intensity is discriminable. This represents "good" brightness discrimination. Conversely, a large value of $\Delta I_c / I$ means that a large percentage change in intensity is required for the eye to detect the change. This represents "poor" brightness discrimination.

A plot of $\Delta I_c / I$ as a function of $\log I$ has the characteristic shape shown in Fig. 2.6. This curve shows that brightness discrimination is poor (the Weber ratio is large) at low levels of illumination, and it improves significantly (the Weber ratio decreases) as background illumination increases. The two branches in the curve reflect the fact that at low levels of illumination vision is carried out by the rods, whereas, at high levels, vision is a function of cones.

If the background illumination is held constant and the intensity of the other source, instead of flashing, is now allowed to vary incrementally from never being perceived to always being perceived, the typical observer can discern a total of one to two dozen different intensity changes. Roughly, this result is related to the number of different intensities a person can see at any one *point* or *small area* in a monochrome image. This does not mean that an image can be represented by such a small number of intensity values because, as the eye roams about the image, the average

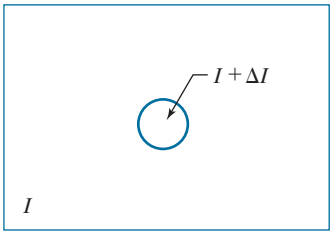



**FIGURE 2.5** Basic experimental setup used to characterize brightness discrimination.

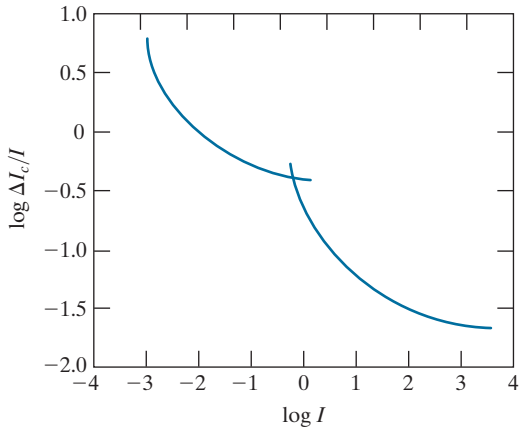



**FIGURE 2.6**
A typical plot of the Weber ratio as a function of intensity.

background changes, thus allowing a *different* set of incremental changes to be detected at each new adaptation level. The net result is that the eye is capable of a broader range of *overall* intensity discrimination. In fact, as we will show in Section 2.4, the eye is capable of detecting objectionable effects in monochrome images whose overall intensity is represented by fewer than approximately two dozen levels.

Two phenomena demonstrate that perceived brightness is not a simple function of intensity. The first is based on the fact that the visual system tends to undershoot or overshoot around the boundary of regions of different intensities. Figure 2.7(a) shows a striking example of this phenomenon. Although the intensity of the stripes

a
b
c

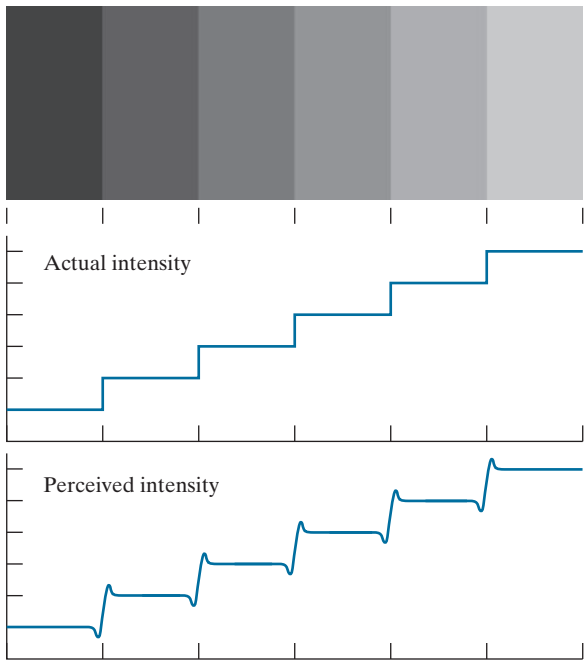



**FIGURE 2.7**
Illustration of the Mach band effect. Perceived intensity is not a simple function of actual intensity.

a b c

**FIGURE 2.8** Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

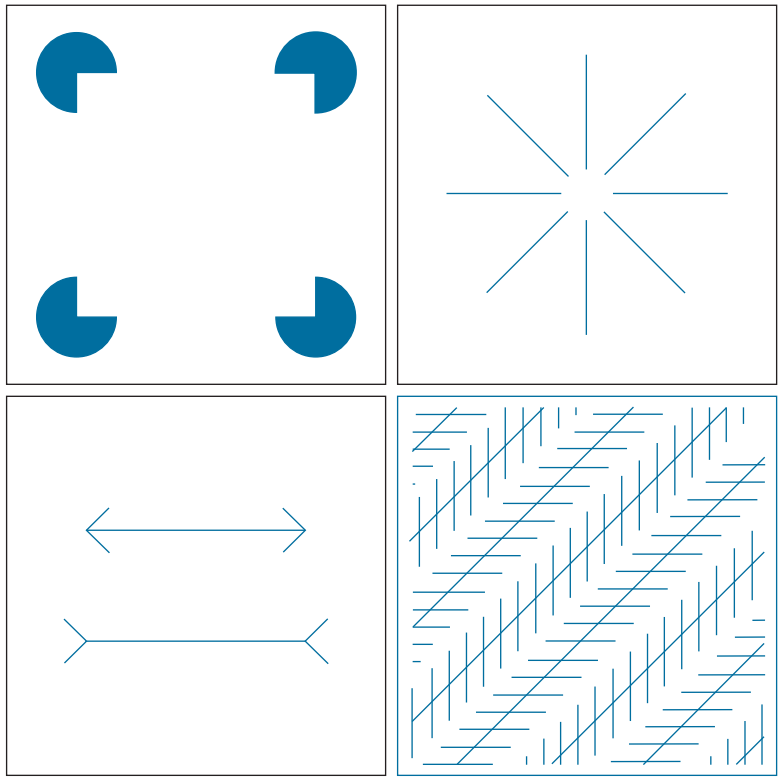
is constant [see Fig. 2.7(b)], we actually perceive a brightness pattern that is strongly scalloped near the boundaries, as Fig. 2.7(c) shows. These perceived scalloped bands are called *Mach bands* after Ernst Mach, who first described the phenomenon in 1865.

The second phenomenon, called *simultaneous contrast*, is that a region's perceived brightness does not depend only on its intensity, as Fig. 2.8 demonstrates. All the center squares have exactly the same intensity, but each appears to the eye to become darker as the background gets lighter. A more familiar example is a piece of paper that looks white when lying on a desk, but can appear totally black when used to shield the eyes while looking directly at a bright sky.

Other examples of human perception phenomena are *optical illusions*, in which the eye fills in nonexisting details or wrongly perceives geometrical properties of objects. Figure 2.9 shows some examples. In Fig. 2.9(a), the outline of a square is seen clearly, despite the fact that no lines defining such a figure are part of the image. The same effect, this time with a circle, can be seen in Fig. 2.9(b); note how just a few lines are sufficient to give the illusion of a complete circle. The two horizontal line segments in Fig. 2.9(c) are of the same length, but one appears shorter than the other. Finally, all long lines in Fig. 2.9(d) are equidistant and parallel. Yet, the crosshatching creates the illusion that those lines are far from being parallel.

## 2.2 LIGHT AND THE ELECTROMAGNETIC SPECTRUM

The electromagnetic spectrum was introduced in Section 1.3. We now consider this topic in more detail. In 1666, Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging beam of light is not white but consists instead of a continuous spectrum of colors ranging from violet at one end to red at the other. As Fig. 2.10 shows, the range of colors we perceive in visible light is a small portion of the electromagnetic spectrum. On one end of the spectrum are radio waves with wavelengths billions of times longer than those of visible light. On the other end of the spectrum are gamma rays with wavelengths millions of times smaller than those of visible light. We showed examples in Section 1.3 of images in most of the bands in the EM spectrum.



a b
c d

**FIGURE 2.9** Some well-known optical illusions.

The electromagnetic spectrum can be expressed in terms of wavelength, frequency, or energy. Wavelength ($\lambda$) and frequency ($\nu$) are related by the expression

$$\lambda = \frac{c}{\nu} \tag{2-1}$$

where $c$ is the speed of light ($2.998 \times 10^8$ m/s). Figure 2.11 shows a schematic representation of one wavelength.

The energy of the various components of the electromagnetic spectrum is given by the expression

$$E = h\nu \tag{2-2}$$

where $h$ is Planck's constant. The units of wavelength are meters, with the terms *microns* (denoted $\mu$m and equal to $10^{-6}$ m) and *nanometers* (denoted nm and equal to $10^{-9}$ m) being used just as frequently. Frequency is measured in *Hertz* (Hz), with one Hz being equal to one cycle of a sinusoidal wave per second. A commonly used unit of energy is the *electron-volt*.

Electromagnetic waves can be visualized as propagating sinusoidal waves with wavelength $\lambda$ (Fig. 2.11), or they can be thought of as a stream of massless particles,

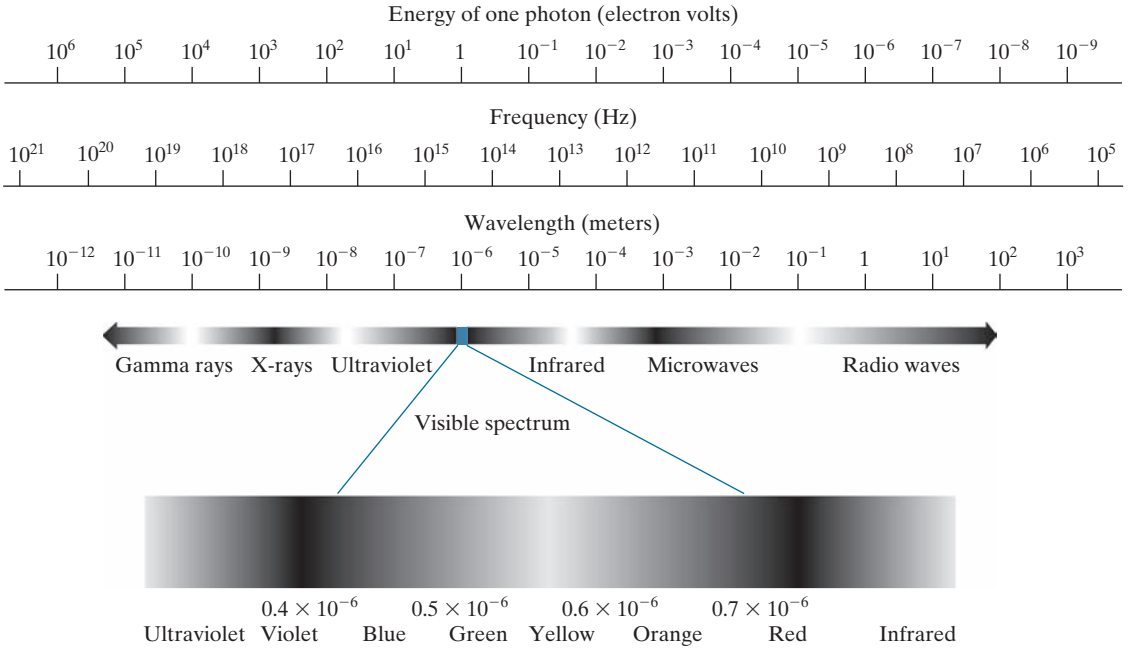



**FIGURE 2.10** The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanations, but note that it encompasses a very narrow range of the total EM spectrum.

each traveling in a wavelike pattern and moving at the speed of light. Each massless particle contains a certain amount (or bundle) of energy, called a *photon*. We see from Eq. (2-2) that energy is proportional to frequency, so the higher-frequency (shorter wavelength) electromagnetic phenomena carry more energy per photon. Thus, radio waves have photons with low energies, microwaves have more energy than radio waves, infrared still more, then visible, ultraviolet, X-rays, and finally gamma rays, the most energetic of all. High-energy electromagnetic radiation, especially in the X-ray and gamma ray bands, is particularly harmful to living organisms.

Light is a type of electromagnetic radiation that can be sensed by the eye. The visible (color) spectrum is shown expanded in Fig. 2.10 for the purpose of discussion (we will discuss color in detail in Chapter 6). The visible band of the electromagnetic spectrum spans the range from approximately 0.43 $\mu$m (violet) to about 0.79 $\mu$m (red). For convenience, the color spectrum is divided into six broad regions: violet, blue, green, yellow, orange, and red. No color (or other component of the

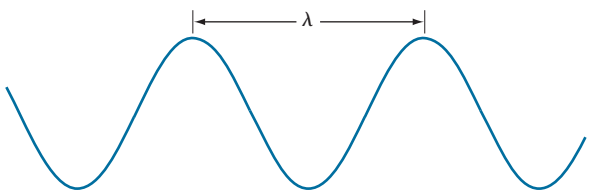



**FIGURE 2.11** Graphical representation of one wavelength.

electromagnetic spectrum) ends abruptly; rather, each range blends smoothly into the next, as Fig. 2.10 shows.

The colors perceived in an object are determined by the nature of the light *reflected* by the object. A body that reflects light relatively balanced in all visible wavelengths appears white to the observer. However, a body that favors reflectance in a limited range of the visible spectrum exhibits some shades of color. For example, green objects reflect light with wavelengths primarily in the 500 to 570 nm range, while absorbing most of the energy at other wavelengths.

Light that is void of color is called *monochromatic* (or *achromatic*) light. The only attribute of monochromatic light is its intensity. Because the intensity of monochromatic light is perceived to vary from black to grays and finally to white, the term *gray level* is used commonly to denote monochromatic intensity (we use the terms *intensity* and *gray level* interchangeably in subsequent discussions). The range of values of monochromatic light from black to white is usually called the *gray scale*, and monochromatic images are frequently referred to as *grayscale images*.

*Chromatic* (color) light spans the electromagnetic energy spectrum from approximately 0.43 to 0.79 $\mu$m, as noted previously. In addition to frequency, three other quantities are used to describe a chromatic light source: radiance, luminance, and brightness. *Radiance* is the total amount of energy that flows from the light source, and it is usually measured in watts (W). *Luminance*, measured in lumens (lm), gives a measure of the amount of energy an observer *perceives* from a light source. For example, light emitted from a source operating in the far infrared region of the spectrum could have significant energy (radiance), but an observer would hardly perceive it; its luminance would be almost zero. Finally, as discussed in Section 2.1, *brightness* is a subjective descriptor of light perception that is practically impossible to measure. It embodies the achromatic notion of intensity and is one of the key factors in describing color sensation.

In principle, if a sensor can be developed that is capable of detecting energy radiated in a band of the electromagnetic spectrum, we can image events of interest in that band. Note, however, that the wavelength of an electromagnetic wave required to "see" an object must be of the same size as, or smaller than, the object. For example, a water molecule has a diameter on the order of $10^{-10}$ m. Thus, to study these molecules, we would need a source capable of emitting energy in the far (high-energy) ultraviolet band or soft (low-energy) X-ray bands.

Although imaging is based predominantly on energy from electromagnetic wave radiation, this is not the only method for generating images. For example, we saw in Section 1.3 that sound reflected from objects can be used to form ultrasonic images. Other sources of digital images are electron beams for electron microscopy, and software for generating synthetic images used in graphics and visualization.

## 2.3 IMAGE SENSING AND ACQUISITION

Most of the images in which we are interested are generated by the combination of an "illumination" source and the reflection or absorption of energy from that source by the elements of the "scene" being imaged. We enclose *illumination* and *scene* in quotes to emphasize the fact that they are considerably more general than the

familiar situation in which a visible light source illuminates a familiar 3-D scene. For example, the illumination may originate from a source of electromagnetic energy, such as a radar, infrared, or X-ray system. But, as noted earlier, it could originate from less traditional sources, such as ultrasound or even a computer-generated illumination pattern. Similarly, the scene elements could be familiar objects, but they can just as easily be molecules, buried rock formations, or a human brain. Depending on the nature of the source, illumination energy is reflected from, or transmitted through, objects. An example in the first category is light reflected from a planar surface. An example in the second category is when X-rays pass through a patient's body for the purpose of generating a diagnostic X-ray image. In some applications, the reflected or transmitted energy is focused onto a photo converter (e.g., a phosphor screen) that converts the energy into visible light. Electron microscopy and some applications of gamma imaging use this approach.

Figure 2.12 shows the three principal sensor arrangements used to transform incident energy into digital images. The idea is simple: Incoming energy is transformed into a voltage by a combination of the input electrical power and sensor material that is responsive to the type of energy being detected. The output voltage waveform is the response of the sensor, and a digital quantity is obtained by digitizing that response. In this section, we look at the principal modalities for image sensing and generation. We will discuss image digitizing in Section 2.4.

## IMAGE ACQUISITION USING A SINGLE SENSING ELEMENT

Figure 2.12(a) shows the components of a single sensing element. A familiar sensor of this type is the photodiode, which is constructed of silicon materials and whose output is a voltage proportional to light intensity. Using a filter in front of a sensor improves its selectivity. For example, an optical green-transmission filter favors light in the green band of the color spectrum. As a consequence, the sensor output would be stronger for green light than for other visible light components.

In order to generate a 2-D image using a single sensing element, there has to be relative displacements in both the $x$- and $y$-directions between the sensor and the area to be imaged. Figure 2.13 shows an arrangement used in high-precision scanning, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The sensor is mounted on a lead screw that provides motion in the perpendicular direction. A light source is contained inside the drum. As the light passes through the film, its intensity is modified by the film density before it is captured by the sensor. This "modulation" of the light intensity causes corresponding variations in the sensor voltage, which are ultimately converted to image intensity levels by digitization.

This method is an inexpensive way to obtain high-resolution images because mechanical motion can be controlled with high precision. The main disadvantages of this method are that it is slow and not readily portable. Other similar mechanical arrangements use a flat imaging bed, with the sensor moving in two linear directions. These types of mechanical digitizers sometimes are referred to as *transmission microdensitometers*. Systems in which light is reflected from the medium, instead of passing through it, are called *reflection microdensitometers*. Another example of imaging with a single sensing element places a laser source coincident with the

a
b
c

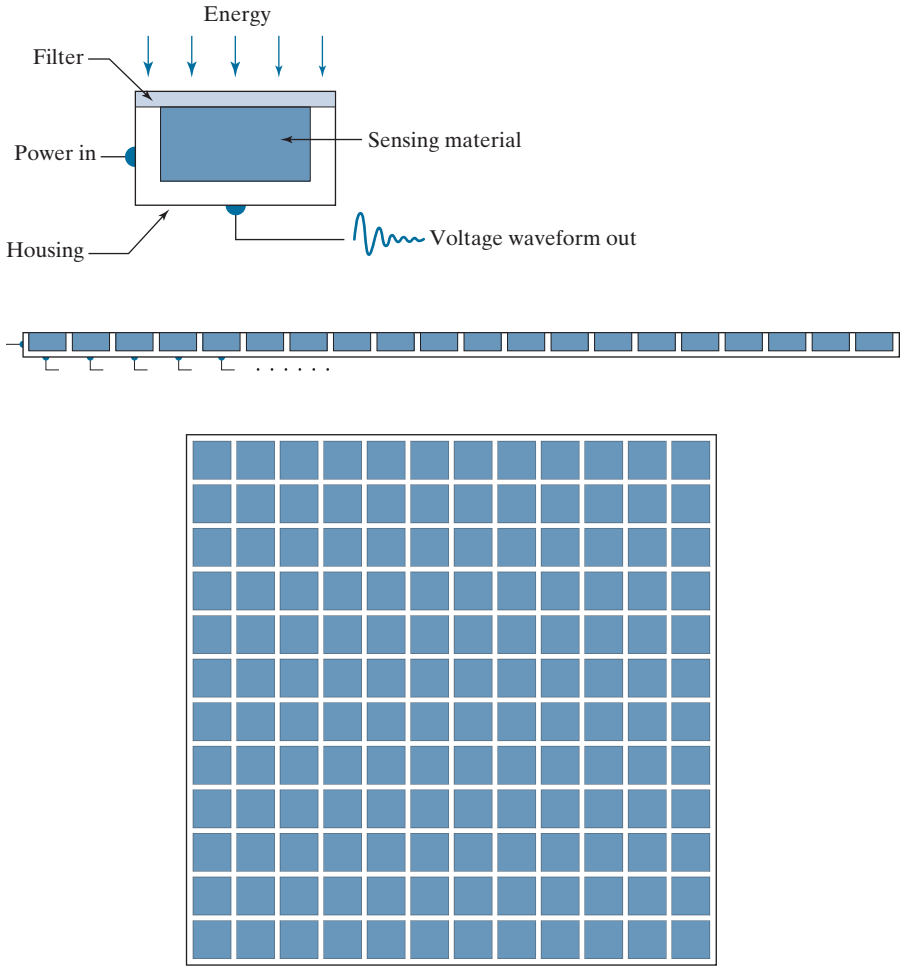



**FIGURE 2.12**
(a) Single sensing element.
(b) Line sensor.
(c) Array sensor.

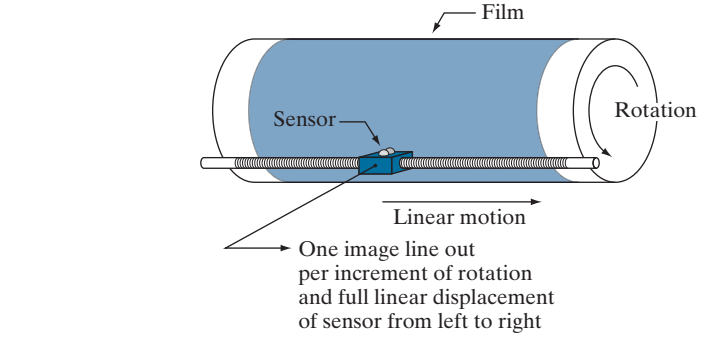



**FIGURE 2.13**
Combining a single sensing element with mechanical motion to generate a 2-D image.

sensor. Moving mirrors are used to control the outgoing beam in a scanning pattern and to direct the reflected laser signal onto the sensor.

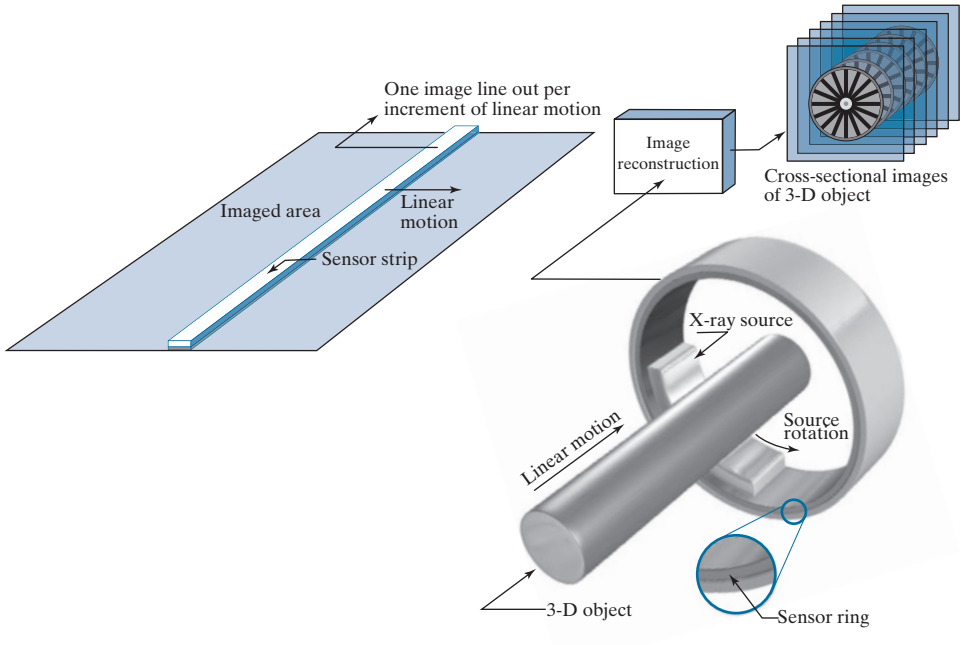
## IMAGE ACQUISITION USING SENSOR STRIPS

A geometry used more frequently than single sensors is an in-line sensor strip, as in Fig. 2.12(b). The strip provides imaging elements in one direction. Motion perpendicular to the strip provides imaging in the other direction, as shown in Fig. 2.14(a). This arrangement is used in most flat bed scanners. Sensing devices with 4000 or more in-line sensors are possible. In-line sensors are used routinely in airborne imaging applications, in which the imaging system is mounted on an aircraft that flies at a constant altitude and speed over the geographical area to be imaged. One-dimensional imaging sensor strips that respond to various bands of the electromagnetic spectrum are mounted perpendicular to the direction of flight. An imaging strip gives one line of an image at a time, and the motion of the strip relative to the scene completes the other dimension of a 2-D image. Lenses or other focusing schemes are used to project the area to be scanned onto the sensors.

Sensor strips in a ring configuration are used in medical and industrial imaging to obtain cross-sectional ("slice") images of 3-D objects, as Fig. 2.14(b) shows. A rotating X-ray source provides illumination, and X-ray sensitive sensors opposite the source collect the energy that passes through the object. This is the basis for medical and industrial computerized axial tomography (CAT) imaging, as indicated in Sections 1.2 and 1.3. The output of the sensors is processed by reconstruction algorithms whose objective is to transform the sensed data into meaningful cross-sectional images (see Section 5.11). In other words, images are not obtained directly

a b

**FIGURE 2.14**
(a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.



from the sensors by motion alone; they also require extensive computer processing. A 3-D digital volume consisting of stacked images is generated as the object is moved in a direction perpendicular to the sensor ring. Other modalities of imaging based on the CAT principle include magnetic resonance imaging (MRI) and positron emission tomography (PET). The illumination sources, sensors, and types of images are different, but conceptually their applications are very similar to the basic imaging approach shown in Fig. 2.14(b).

## IMAGE ACQUISITION USING SENSOR ARRAYS

Figure 2.12(c) shows individual sensing elements arranged in the form of a 2-D array. Electromagnetic and ultrasonic sensing devices frequently are arranged in this manner. This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD (charge-coupled device) array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged arrays of $4000 \times 4000$ elements or more. CCD sensors are used widely in digital cameras and other light-sensing instruments. The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. Noise reduction is achieved by letting the sensor integrate the input light signal over minutes or even hours. Because the sensor array in Fig. 2.12(c) is two-dimensional, its key advantage is that a complete image can be obtained by focusing the energy pattern onto the surface of the array. Motion obviously is not necessary, as is the case with the sensor arrangements discussed in the preceding two sections.

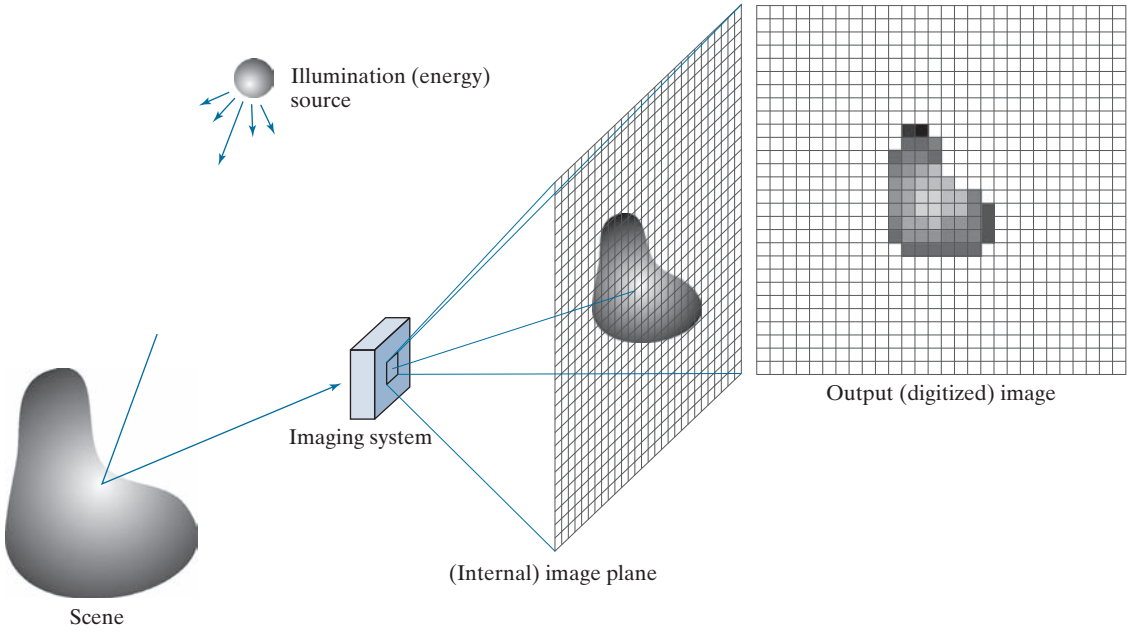
In some cases, the source is imaged directly, as in obtaining images of the sun.

Figure 2.15 shows the principal manner in which array sensors are used. This figure shows the energy from an illumination source being reflected from a scene (as mentioned at the beginning of this section, the energy also could be transmitted through the scene). The first function performed by the imaging system in Fig. 2.15(c) is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is an optical lens that projects the viewed scene onto the focal plane of the lens, as Fig. 2.15(d) shows. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and convert them to an analog signal, which is then digitized by another section of the imaging system. The output is a digital image, as shown diagrammatically in Fig. 2.15(e). Converting images into digital form is the topic of Section 2.4.

## A SIMPLE IMAGE FORMATION MODEL

As introduced in Section 1.1, we denote images by two-dimensional functions of the form $f(x, y)$. The value of $f$ at spatial coordinates $(x, y)$ is a scalar quantity whose physical meaning is determined by the source of the image, and whose values are proportional to energy radiated by a physical source (e.g., electromagnetic waves). As a consequence, $f(x, y)$ must be nonnegative[†] and finite; that is,

[†] Image intensities can become negative during processing, or as a result of interpretation. For example, in radar images, objects moving toward the radar often are interpreted as having negative velocities while objects moving away are interpreted as having positive velocities. Thus, a velocity image might be coded as having both positive and negative values. When storing and displaying images, we normally scale the intensities so that the smallest negative value becomes 0 (see Section 2.6 regarding intensity scaling).




a
b c d e

**FIGURE 2.15** An example of digital image acquisition. (a) Illumination (energy) source. (b) A scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

$$0 \leq f(x,y) < \infty \tag{2-3}$$

Function $f(x,y)$ is characterized by two components: (1) the amount of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene. Appropriately, these are called the *illumination* and *reflectance* components, and are denoted by $i(x,y)$ and $r(x,y)$, respectively. The two functions combine as a product to form $f(x,y)$:

$$f(x,y) = i(x,y)r(x,y) \tag{2-4}$$

where

$$0 \leq i(x,y) < \infty \tag{2-5}$$

and

$$0 \leq r(x,y) \leq 1 \tag{2-6}$$

Thus, reflectance is bounded by 0 (total absorption) and 1 (total reflectance). The nature of $i(x,y)$ is determined by the illumination source, and $r(x,y)$ is determined by the characteristics of the imaged objects. These expressions are applicable also to images formed via transmission of the illumination through a medium, such as a

chest X-ray. In this case, we would deal with a *transmissivity* instead of a *reflectivity* function, but the limits would be the same as in Eq. (2-6), and the image function formed would be modeled as the product in Eq. (2-4).

**EXAMPLE 2.1: Some typical values of illumination and reflectance.**

The following numerical quantities illustrate some typical values of illumination and reflectance for visible light. On a clear day, the sun may produce in excess of 90,000 lm/m$^2$ of illumination on the surface of the earth. This value decreases to less than 10,000 lm/m$^2$ on a cloudy day. On a clear evening, a full moon yields about 0.1 lm/m$^2$ of illumination. The typical illumination level in a commercial office is about 1,000 lm/m$^2$. Similarly, the following are typical values of $r(x, y)$: 0.01 for black velvet, 0.65 for stainless steel, 0.80 for flat-white wall paint, 0.90 for silver-plated metal, and 0.93 for snow.

Let the intensity (gray level) of a monochrome image at any coordinates $(x, y)$ be denoted by

$$\ell = f(x, y) \tag{2-7}$$

From Eqs. (2-4) through (2-6) it is evident that $\ell$ lies in the range

$$L_{\min} \leq \ell \leq L_{\max} \tag{2-8}$$

In theory, the requirement on $L_{\min}$ is that it be nonnegative, and on $L_{\max}$ that it be finite. In practice, $L_{\min} = i_{\min} r_{\min}$ and $L_{\max} = i_{\max} r_{\max}$. From Example 2.1, using average office illumination and reflectance values as guidelines, we may expect $L_{\min} \approx 10$ and $L_{\max} \approx 1000$ to be typical indoor values in the absence of additional illumination. The units of these quantities are lum/m$^2$. However, actual units seldom are of interest, except in cases where photometric measurements are being performed.

The interval $[L_{\min}, L_{\max}]$ is called the *intensity* (or *gray*) *scale*. Common practice is to shift this interval numerically to the interval $[0, 1]$, or $[0, C]$, where $\ell = 0$ is considered black and $\ell = 1$ (or $C$) is considered white on the scale. All intermediate values are shades of gray varying from black to white.

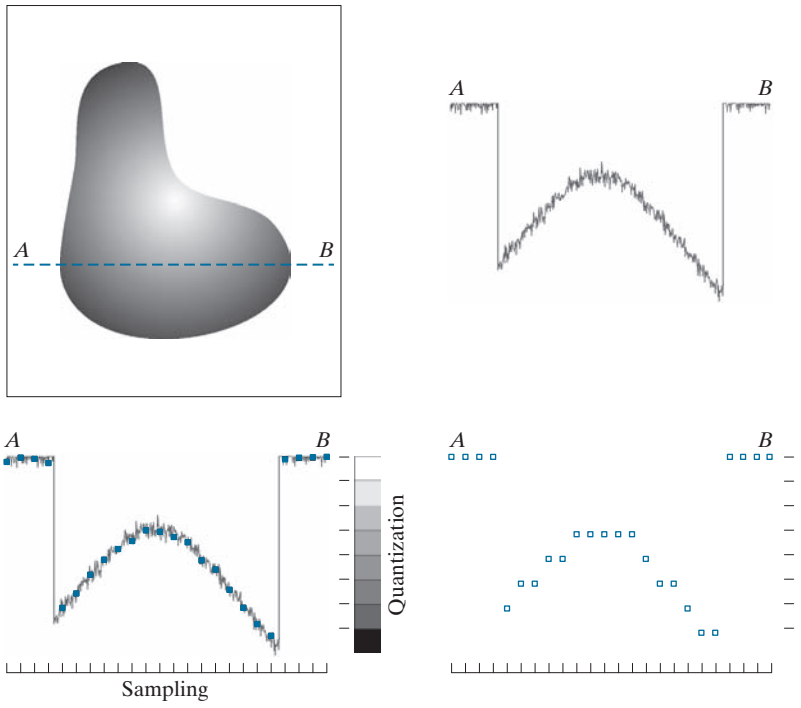
## 2.4 IMAGE SAMPLING AND QUANTIZATION

The discussion of sampling in this section is of an intuitive nature. We will discuss this topic in depth in Chapter 4.

As discussed in the previous section, there are numerous ways to acquire images, but our objective in all is the same: to generate digital images from sensed data. The output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. To create a digital image, we need to convert the continuous sensed data into a digital format. This requires two processes: *sampling* and *quantization*.

### BASIC CONCEPTS IN SAMPLING AND QUANTIZATION

Figure 2.16(a) shows a continuous image $f$ that we want to convert to digital form. An image may be continuous with respect to the $x$- and $y$-coordinates, and also in




a b
c d

**FIGURE 2.16**
(a) Continuous image. (b) A scan line showing intensity variations along line $AB$ in the continuous image. (c) Sampling and quantization. (d) Digital scan line. (The black border in (a) is included for clarity. It is not part of the image).

amplitude. To digitize it, we have to sample the function in both coordinates and also in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

The one-dimensional function in Fig. 2.16(b) is a plot of amplitude (intensity level) values of the continuous image along the line segment $AB$ in Fig. 2.16(a). The random variations are due to image noise. To sample this function, we take equally spaced samples along line $AB$, as shown in Fig. 2.16(c). The samples are shown as small dark squares superimposed on the function, and their (discrete) spatial locations are indicated by corresponding tick marks in the bottom of the figure. The set of dark squares constitute the *sampled* function. However, the *values* of the samples still span (vertically) a continuous range of intensity values. In order to form a digital function, the intensity values also must be converted (*quantized*) into *discrete* quantities. The vertical gray bar in Fig. 2.16(c) depicts the intensity scale divided into eight discrete intervals, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight intensity intervals. The continuous intensity levels are quantized by assigning one of the eight values to each sample, depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization are shown as white squares in Fig. 2.16(d). Starting at the top of the continuous image and carrying out this procedure downward, line by line, produces a two-dimensional digital image. It is implied in Fig. 2.16 that, in addition to the number of discrete levels used, the accuracy achieved in quantization is highly dependent on the noise content of the sampled signal.

a b

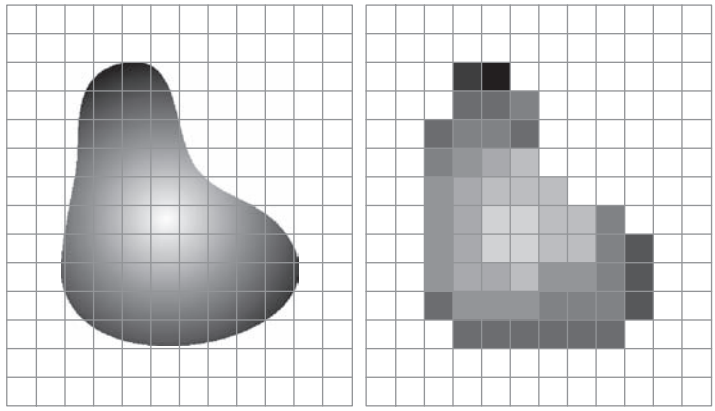


**FIGURE 2.17**
(a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

In practice, the method of sampling is determined by the sensor arrangement used to generate the image. When an image is generated by a single sensing element combined with mechanical motion, as in Fig. 2.13, the output of the sensor is quantized in the manner described above. However, spatial sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data. Mechanical motion can be very exact so, in principle, there is almost no limit on how fine we can sample an image using this approach. In practice, limits on sampling accuracy are determined by other factors, such as the quality of the optical components used in the system.

When a sensing strip is used for image acquisition, the number of sensors in the strip establishes the samples in the resulting image in one direction, and mechanical motion establishes the number of samples in the other. Quantization of the sensor outputs completes the process of generating a digital image.

When a sensing array is used for image acquisition, no motion is required. The number of sensors in the array establishes the limits of sampling in both directions. Quantization of the sensor outputs is as explained above. Figure 2.17 illustrates this concept. Figure 2.17(a) shows a continuous image projected onto the plane of a 2-D sensor. Figure 2.17(b) shows the image after sampling and quantization. The quality of a digital image is determined to a large degree by the number of samples and discrete intensity levels used in sampling and quantization. However, as we will show later in this section, image content also plays a role in the choice of these parameters.

## REPRESENTING DIGITAL IMAGES

Let $f(s, t)$ represent a *continuous* image function of two continuous variables, $s$ and $t$. We convert this function into a *digital image* by sampling and quantization, as explained in the previous section. Suppose that we sample the continuous image into a digital image, $f(x, y)$, containing $M$ rows and $N$ columns, where $(x, y)$ are discrete coordinates. For notational clarity and convenience, we use integer values for these discrete coordinates: $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. Thus, for example, the value of the digital image at the origin is $f(0,0)$, and its value at the next coordinates along the first row is $f(0,1)$. Here, the notation (0, 1) is used

to denote the second sample along the first row. It *does not* mean that these are the values of the physical coordinates when the image was sampled. In general, the value of a digital image at any coordinates $(x, y)$ is denoted $f(x, y)$, where $x$ and $y$ are integers. When we need to refer to specific coordinates $(i, j)$, we use the notation $f(i, j)$, where the arguments are integers. The section of the real plane spanned by the coordinates of an image is called the *spatial domain*, with $x$ and $y$ being referred to as *spatial variables* or *spatial coordinates*.

Figure 2.18 shows three ways of representing $f(x, y)$. Figure 2.18(a) is a plot of the function, with two axes determining spatial location and the third axis being the values of $f$ as a function of $x$ and $y$. This representation is useful when working with grayscale sets whose elements are expressed as triplets of the form $(x, y, z)$, where $x$ and $y$ are spatial coordinates and $z$ is the value of $f$ at coordinates $(x, y)$. We will work with this representation briefly in Section 2.6.

The representation in Fig. 2.18(b) is more common, and it shows $f(x, y)$ as it would appear on a computer display or photograph. Here, the intensity of each point in the display is proportional to the value of $f$ at that point. In this figure, there are only three equally spaced intensity values. If the intensity is normalized to the interval $[0,1]$, then each point in the image has the value $0, 0.5$, or $1$. A monitor or printer converts these three values to black, gray, or white, respectively, as in Fig. 2.18(b). This type of representation includes color images, and allows us to view results at a glance.

As Fig. 2.18(c) shows, the third representation is an array (matrix) composed of the numerical values of $f(x, y)$. This is the representation used for computer processing. In equation form, we write the representation of an $M \times N$ numerical array as

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2-9}$$

The right side of this equation is a digital image represented as an array of real numbers. Each element of this array is called an *image element*, *picture element*, *pixel*, or *pel*. We use the terms *image* and *pixel* throughout the book to denote a digital image and its elements. Figure 2.19 shows a graphical representation of an image array, where the $x$- and $y$-axis are used to denote the rows and columns of the array. Specific pixels are values of the array at a fixed pair of coordinates. As mentioned earlier, we generally use $f(i, j)$ when referring to a pixel with coordinates $(i, j)$.

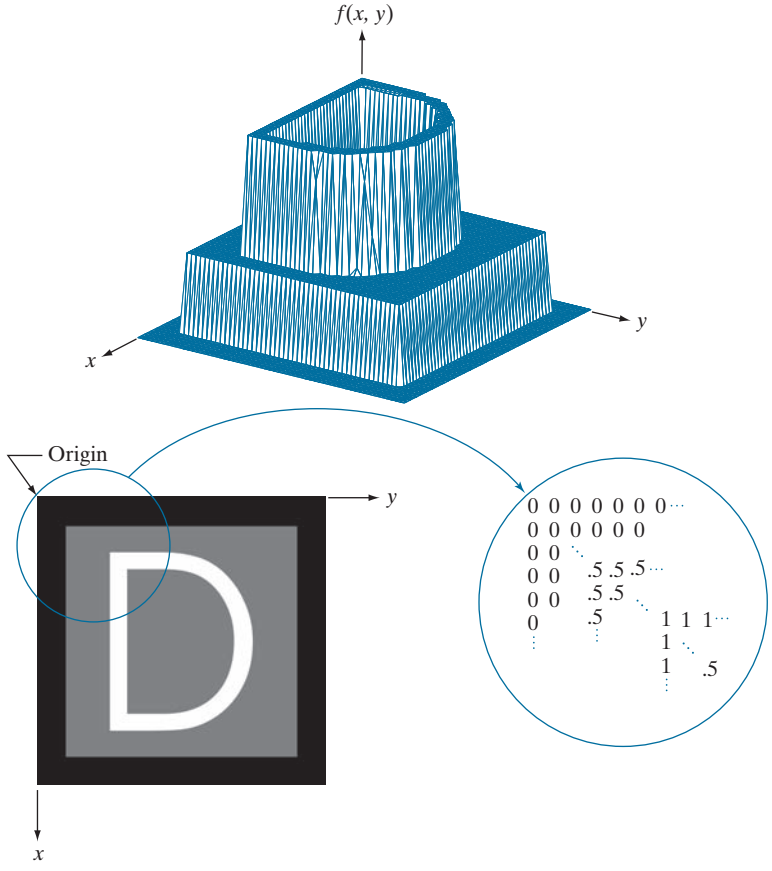
We can also represent a digital image in a traditional matrix form:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \tag{2-10}$$

Clearly, $a_{ij} = f(i, j)$, so Eqs. (2-9) and (2-10) denote identical arrays.

a
b c

**FIGURE 2.18**
(a) Image plotted as a surface. (b) Image displayed as a visual intensity array. (c) Image shown as a 2-D numerical array. (The numbers 0, .5, and 1 represent black, gray, and white, respectively.)




As Fig. 2.19 shows, we define the *origin* of an image at the top left corner. This is a convention based on the fact that many image displays (e.g., TV monitors) sweep an image starting at the top left and moving to the right, one row at a time. More important is the fact that the first element of a matrix is by convention at the top left of the array. Choosing the origin of $f(x, y)$ at that point makes sense mathematically because digital images in reality are matrices. In fact, as you will see, sometimes we use $x$ and $y$ interchangeably in equations with the *rows* ($r$) and *columns* ($c$) of a matrix.

It is important to note that the representation in Fig. 2.19, in which the positive $x$-axis extends downward and the positive $y$-axis extends to the right, is precisely the right-handed Cartesian coordinate system with which you are familiar,[†] but shown rotated by 90° so that the origin appears on the top, left.

[†]Recall that a right-handed coordinate system is such that, when the index of the right hand points in the direction of the positive $x$-axis and the middle finger points in the (perpendicular) direction of the positive $y$-axis, the thumb points up. As Figs. 2.18 and 2.19 show, this indeed is the case in our image coordinate system. In practice, you will also find implementations based on a left-handed system, in which the $x$- and $y$-axis are interchanged from the way we show them in Figs. 2.18 and 2.19. For example, MATLAB uses a left-handed system for image processing. Both systems are perfectly valid, provided they are used consistently.

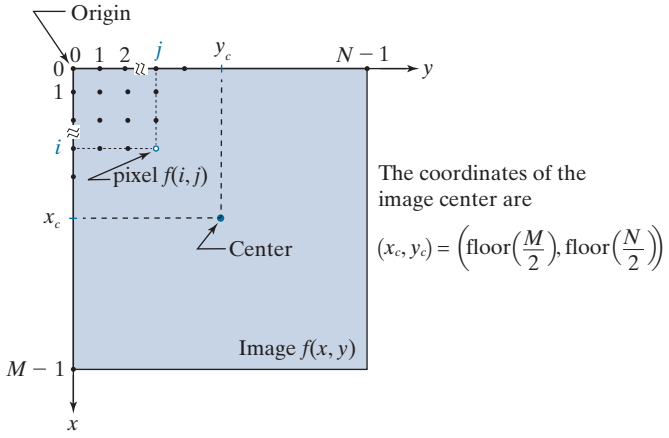



**FIGURE 2.19** Coordinate convention used to represent digital images. Because coordinate values are integers, there is a one-to-one correspondence between $x$ and $y$ and the rows ($r$) and columns ($c$) of a matrix.

The *center* of an $M \times N$ digital image with origin at $(0,0)$ and range to $(M-1, N-1)$ is obtained by dividing $M$ and $N$ by 2 and rounding *down* to the nearest integer. This operation sometimes is denoted using the floor operator, $\lfloor \cdot \rfloor$, as shown in Fig. 2.19. This holds true for $M$ and $N$ even *or* odd. For example, the center of an image of size $1023 \times 1024$ is at (511, 512). Some programming languages (e.g., MATLAB) start indexing at 1 instead of at 0. The center of an image in that case is found at $(x_c, y_c) = (\text{floor}(M/2) + 1, \text{floor}(N/2) + 1)$.

The *floor* of $z$, sometimes denoted $\lfloor z \rfloor$, is the largest integer that is less than or equal to $z$. The *ceiling* of $z$, denoted $\lceil z \rceil$, is the smallest integer that is greater than or equal to $z$.

To express sampling and quantization in more formal mathematical terms, let $Z$ and $R$ denote the set of integers and the set of real numbers, respectively. The sampling process may be viewed as partitioning the $xy$-plane into a grid, with the coordinates of the center of each cell in the grid being a pair of elements from the Cartesian product $Z^2$ (also denoted $Z \times Z$) which, as you may recall, is the set of all ordered pairs of elements $(z_i, z_j)$ with $z_i$ and $z_j$ being integers from set $Z$. Hence, $f(x, y)$ is a digital image if $(x, y)$ are integers from $Z^2$ and $f$ is a function that assigns an intensity value (that is, a real number from the set of real numbers, $R$) to each distinct pair of coordinates $(x, y)$. This functional assignment is the quantization process described earlier. If the intensity levels also are integers, then $R = Z$, and a digital image becomes a 2-D function whose coordinates and amplitude values are integers. This is the representation we use in the book.

See Eq. (2-41) in Section 2.6 for a formal definition of the Cartesian product.

Image digitization requires that decisions be made regarding the values for $M$, $N$, and for the number, $L$, of discrete intensity levels. There are no restrictions placed on $M$ and $N$, other than they have to be positive integers. However, digital storage and quantizing hardware considerations usually lead to the number of intensity levels, $L$, being an integer power of two; that is

$$L = 2^k \tag{2-11}$$

where $k$ is an integer. We assume that the discrete levels are equally spaced and that they are integers in the range $[0, L-1]$.

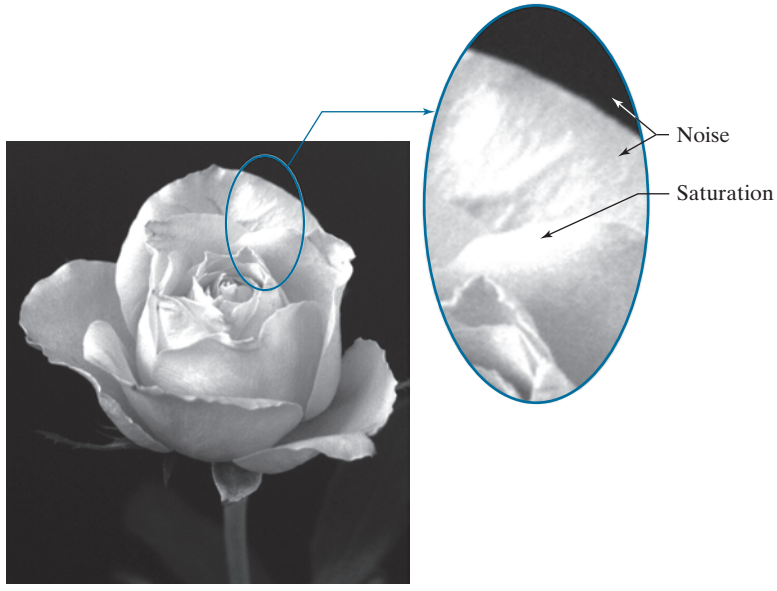



**FIGURE 2.20**
An image exhibiting saturation and noise. Saturation is the highest value beyond which all intensity values are clipped (note how the entire saturated area has a high, constant intensity level). Visible noise in this case appears as a grainy texture pattern. The dark background is noisier, but the noise is difficult to see.

Sometimes, the range of values spanned by the gray scale is referred to as the *dynamic range*, a term used in different ways in different fields. Here, we define the dynamic range of an imaging system to be the ratio of the maximum measurable intensity to the minimum detectable intensity level in the system. As a rule, the upper limit is determined by *saturation* and the lower limit by *noise*, although noise can be present also in lighter intensities. Figure 2.20 shows examples of saturation and slight visible noise. Because the darker regions are composed primarily of pixels with the minimum detectable intensity, the background in Fig. 2.20 is the noisiest part of the image; however, dark background noise typically is much harder to see.

The dynamic range establishes the lowest and highest intensity levels that a system can represent and, consequently, that an image can have. Closely associated with this concept is *image contrast*, which we define as the difference in intensity between the highest and lowest intensity levels in an image. The *contrast ratio* is the ratio of these two quantities. When an appreciable number of pixels in an image have a high dynamic range, we can expect the image to have high contrast. Conversely, an image with low dynamic range typically has a dull, washed-out gray look. We will discuss these concepts in more detail in Chapter 3.

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

$$b = M \times N \times k \tag{2-12}$$

When $M = N$, this equation becomes

$$b = N^2 k \tag{2-13}$$

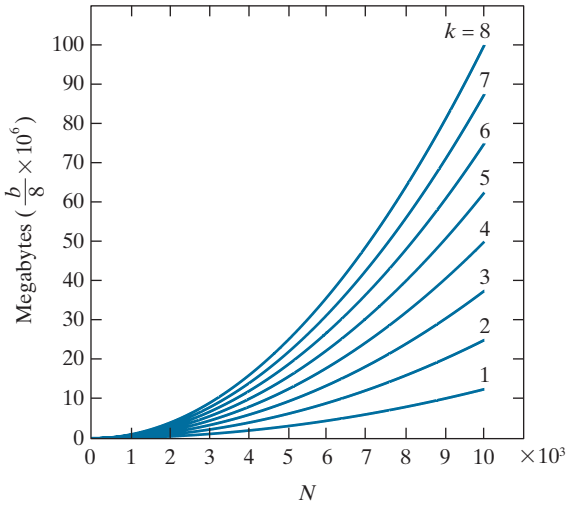



**FIGURE 2.21** Number of megabytes required to store images for various values of $N$ and $k$.

Figure 2.21 shows the number of megabytes required to store square images for various values of $N$ and $k$ (as usual, one byte equals 8 bits and a megabyte equals $10^6$ bytes).

When an image can have $2^k$ possible intensity levels, it is common practice to refer to it as a "$k$-bit image," (e,g., a 256-level image is called an *8-bit image*). Note that storage requirements for large 8-bit images (e.g., $10,000 \times 10,000$ pixels) are not insignificant.

## LINEAR VS. COORDINATE INDEXING

The convention discussed in the previous section, in which the location of a pixel is given by its 2-D coordinates, is referred to as *coordinate indexing*, or *subscript indexing*. Another type of indexing used extensively in programming image processing algorithms is *linear indexing*, which consists of a 1-D string of nonnegative integers based on computing offsets from coordinates $(0,0)$. There are two principal types of linear indexing, one is based on a row scan of an image, and the other on a column scan.

Figure 2.22 illustrates the principle of linear indexing based on a column scan. The idea is to scan an image column by column, starting at the origin and proceeding down and then to the right. The linear index is based on counting pixels as we scan the image in the manner shown in Fig. 2.22. Thus, a scan of the first (leftmost) column yields linear indices 0 through $M-1$. A scan of the second column yields indices $M$ through $2M-1$, and so on, until the last pixel in the last column is assigned the linear index value $MN-1$. Thus, a linear index, denoted by $\alpha$, has one of $MN$ possible values: $0, 1, 2, \ldots, MN-1$, as Fig. 2.22 shows. The important thing to notice here is that each pixel is assigned a linear index value that identifies it uniquely.

The formula for generating linear indices based on a column scan is straightforward and can be determined by inspection. For any pair of coordinates $(x, y)$, the corresponding linear index value is

$$\alpha = My + x \tag{2-14}$$

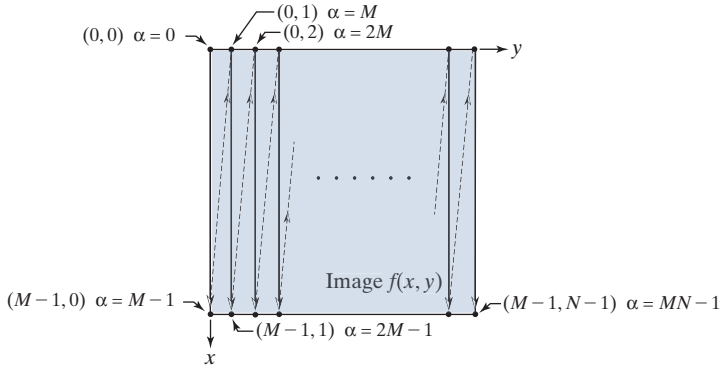



**FIGURE 2.22** Illustration of column scanning for generating linear indices. Shown are several 2-D coordinates (in parentheses) and their corresponding linear indices.

Conversely, the coordinate indices for a given linear index value $\alpha$ are given by the equations[†]

$$x = \alpha \bmod M \tag{2-15}$$

and

$$y = (\alpha - x)/M \tag{2-16}$$

Recall that $\alpha \bmod M$ means "the remainder of the division of $\alpha$ by $M$." This is a formal way of stating that row numbers repeat themselves at the start of every column. Thus, when $\alpha = 0$, the remainder of the division of 0 by $M$ is 0, so $x = 0$. When $\alpha = 1$, the remainder is 1, and so $x = 1$. You can see that $x$ will continue to be equal to $\alpha$ until $\alpha = M - 1$. When $\alpha = M$ (which is at the beginning of the second column), the remainder is 0, and thus $x = 0$ again, and it increases by 1 until the next column is reached, when the pattern repeats itself. Similar comments apply to Eq. (2-16). See Problem 2.11 for a derivation of the preceding two equations.

## SPATIAL AND INTENSITY RESOLUTION

Intuitively, *spatial resolution* is a measure of the smallest discernible detail in an image. Quantitatively, spatial resolution can be stated in several ways, with *line pairs per unit distance*, and *dots (pixels) per unit distance* being common measures. Suppose that we construct a chart with alternating black and white vertical lines, each of width $W$ units ($W$ can be less than 1). The width of a *line pair* is thus $2W$, and there are $W/2$ line pairs per unit distance. For example, if the width of a line is 0.1 mm, there are 5 line pairs per unit distance (i.e., per mm). A widely used definition of image resolution is the largest number of *discernible* line pairs per unit distance (e.g., 100 line pairs per mm). Dots per unit distance is a measure of image resolution used in the printing and publishing industry. In the U.S., this measure usually is expressed as *dots per inch* (dpi). To give you an idea of quality, newspapers are printed with a

[†]When working with modular number systems, it is more accurate to write $x \equiv \alpha \bmod M$, where the symbol $\equiv$ means *congruence*. However, our interest here is just on converting from linear to coordinate indexing, so we use the more familiar equal sign.

resolution of 75 dpi, magazines at 133 dpi, glossy brochures at 175 dpi, and the book page at which you are presently looking was printed at 2400 dpi.

To be meaningful, measures of spatial resolution must be stated with respect to spatial units. Image size by itself does not tell the complete story. For example, to say that an image has a resolution of $1024 \times 1024$ pixels is not a meaningful statement without stating the spatial dimensions encompassed by the image. Size by itself is helpful only in making comparisons between imaging capabilities. For instance, a digital camera with a 20-megapixel CCD imaging chip can be expected to have a higher capability to resolve detail than an 8-megapixel camera, assuming that both cameras are equipped with comparable lenses and the comparison images are taken at the same distance.

*Intensity resolution* similarly refers to the smallest *discernible* change in intensity level. We have considerable discretion regarding the number of spatial samples (pixels) used to generate a digital image, but this is not true regarding the number of intensity levels. Based on hardware considerations, the number of intensity levels usually is an integer power of two, as we mentioned when discussing Eq. (2-11). The most common number is 8 bits, with 16 bits being used in some applications in which enhancement of specific intensity ranges is necessary. Intensity quantization using 32 bits is rare. Sometimes one finds systems that can digitize the intensity levels of an image using 10 or 12 bits, but these are not as common.

Unlike spatial resolution, which must be based on a per-unit-of-distance basis to be meaningful, it is common practice to refer to the number of bits used to quantize intensity as the "*intensity resolution*." For example, it is common to say that an image whose intensity is quantized into 256 levels has 8 bits of intensity resolution. However, keep in mind that *discernible* changes in intensity are influenced also by noise and saturation values, and by the capabilities of human perception to analyze and interpret details in the context of an entire scene (see Section 2.1). The following two examples illustrate the effects of spatial and intensity resolution on discernible detail. Later in this section, we will discuss how these two parameters interact in determining perceived image quality.

**EXAMPLE 2.2: Effects of reducing the spatial resolution of a digital image.**

Figure 2.23 shows the effects of reducing the spatial resolution of an image. The images in Figs. 2.23(a) through (d) have resolutions of 930, 300, 150, and 72 dpi, respectively. Naturally, the lower resolution images are smaller than the original image in (a). For example, the original image is of size $2136 \times 2140$ pixels, but the 72 dpi image is an array of only $165 \times 166$ pixels. In order to facilitate comparisons, all the smaller images were zoomed back to the original size (the method used for zooming will be discussed later in this section). This is somewhat equivalent to "getting closer" to the smaller images so that we can make comparable statements about visible details.

There are some small visual differences between Figs. 2.23(a) and (b), the most notable being a slight distortion in the seconds marker pointing to 60 on the right side of the chronometer. For the most part, however, Fig. 2.23(b) is quite acceptable. In fact, 300 dpi is the typical minimum image spatial resolution used for book publishing, so one would not expect to see much difference between these two images. Figure 2.23(c) begins to show visible degradation (see, for example, the outer edges of the chronometer

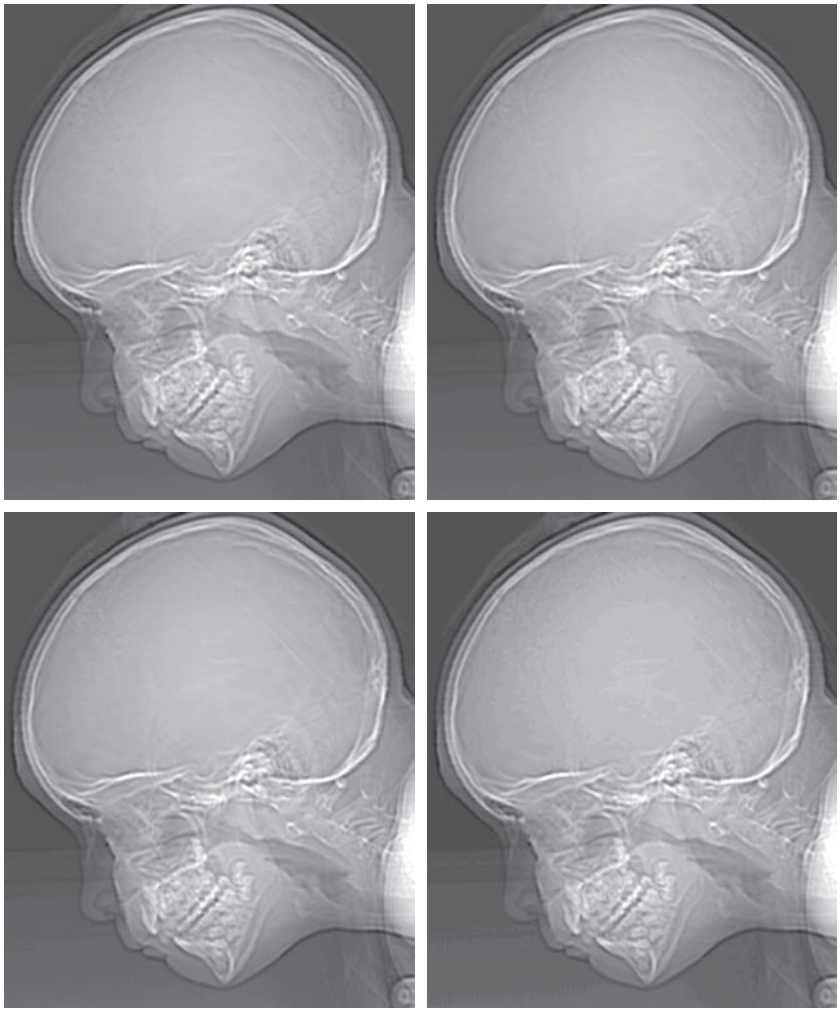
a b
c d

**FIGURE 2.23**
Effects of reducing spatial resolution. The images shown are at:
(a) 930 dpi,
(b) 300 dpi,
(c) 150 dpi, and
(d) 72 dpi.

case and compare the seconds marker with the previous two images). The numbers also show visible degradation. Figure 2.23(d) shows degradation that is visible in most features of the image. When printing at such low resolutions, the printing and publishing industry uses a number of techniques (such as locally varying the pixel size) to produce much better results than those in Fig. 2.23(d). Also, as we will show later in this section, it is possible to improve on the results of Fig. 2.23 by the choice of interpolation method used.

**EXAMPLE 2.3: Effects of varying the number of intensity levels in a digital image.**

Figure 2.24(a) is a $774 \times 640$ CT projection image, displayed using 256 intensity levels (see Chapter 1 regarding CT images). The objective of this example is to reduce the number of intensities of the image from 256 to 2 in integer powers of 2, while keeping the spatial resolution constant. Figures 2.24(b) through (d) were obtained by reducing the number of intensity levels to 128, 64, and 32, respectively (we will discuss in Chapter 3 how to reduce the number of levels).



a b
c d

**FIGURE 2.24**
(a) 774 × 640, 256-level image. (b)-(d) Image displayed in 128, 64, and 32 intensity levels, while keeping the spatial resolution constant. (Original image courtesy of the Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)

The 128- and 64-level images are visually identical for all practical purposes. However, the 32-level image in Fig. 2.24(d) has a set of almost imperceptible, very fine ridge-like structures in areas of constant intensity. These structures are clearly visible in the 16-level image in Fig. 2.24(e). This effect, caused by using an insufficient number of intensity levels in smooth areas of a digital image, is called *false contouring*, so named because the ridges resemble topographic contours in a map. False contouring generally is quite objectionable in images displayed using 16 or fewer uniformly spaced intensity levels, as the images in Figs. 2.24(e)-(h) show.

As a very rough guideline, and assuming integer powers of 2 for convenience, images of size 256 × 256 pixels with 64 intensity levels, and printed on a size format on the order of 5 × 5 cm, are about the lowest spatial and intensity resolution images that can be expected to be reasonably free of objectionable sampling distortions and false contouring.

e f
g h

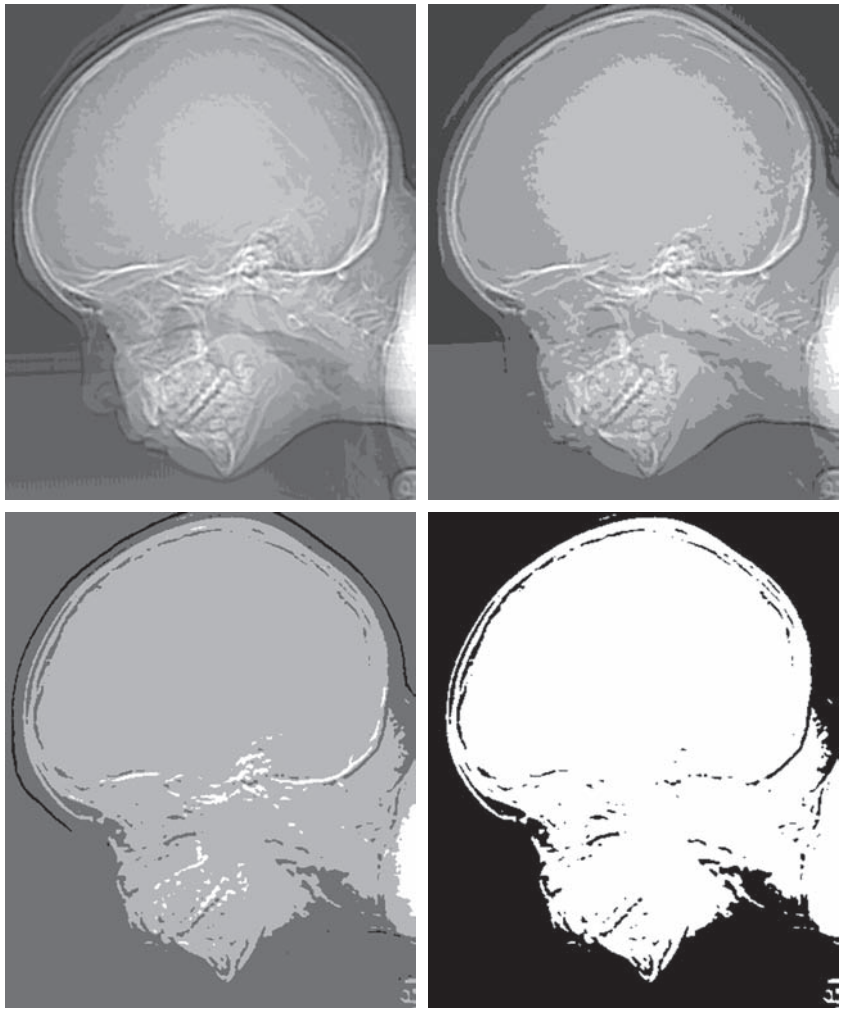


**FIGURE 2.24**
(*Continued*)
(e)-(h) Image displayed in 16, 8, 4, and 2 intensity levels.

The results in Examples 2.2 and 2.3 illustrate the effects produced on image quality by varying spatial and intensity resolution independently. However, these results did not consider any relationships that might exist between these two parameters. An early study by Huang [1965] attempted to quantify experimentally the effects on image quality produced by the interaction of these two variables. The experiment consisted of a set of subjective tests. Images similar to those shown in Fig. 2.25 were used. The woman's face represents an image with relatively little detail; the picture of the cameraman contains an intermediate amount of detail; and the crowd picture contains, by comparison, a large amount of detail.

Sets of these three types of images of various sizes and intensity resolution were generated by varying $N$ and $k$ [see Eq. (2-13)]. Observers were then asked to rank

a b c

**FIGURE 2.25** (a) Image with a low level of detail. (b) Image with a medium level of detail. (c) Image with a relatively large amount of detail. (Image (b) courtesy of the Massachusetts Institute of Technology.)

them according to their subjective quality. Results were summarized in the form of so-called *isopreference curves* in the *Nk*-plane. (Figure 2.26 shows average isopreference curves representative of the types of images in Fig. 2.25.) Each point in the *Nk*-plane represents an image having values of $N$ and $k$ equal to the coordinates of that point. Points lying on an isopreference curve correspond to images of equal subjective quality. It was found in the course of the experiments that the isopreference curves tended to shift right and upward, but their shapes in each of the three image categories were similar to those in Fig. 2.26. These results were not unexpected, because a shift up and right in the curves simply means larger values for $N$ and $k$, which implies better picture quality.

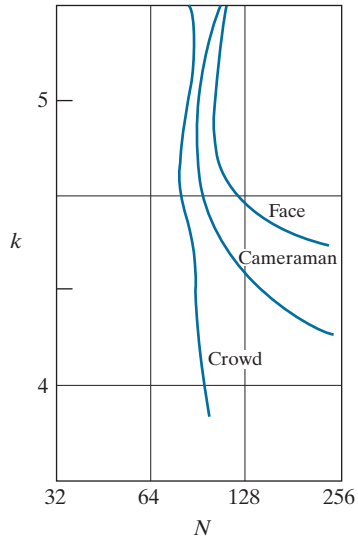



**FIGURE 2.26** Representative isopreference curves for the three types of images in Fig. 2.25.

Observe that isopreference curves tend to become more vertical as the detail in the image increases. This result suggests that for images with a large amount of detail only a few intensity levels may be needed. For example, the isopreference curve in Fig. 2.26 corresponding to the crowd is nearly vertical. This indicates that, for a fixed value of $N$, the perceived quality for this type of image is nearly independent of the number of intensity levels used (for the range of intensity levels shown in Fig. 2.26). The perceived quality in the other two image categories remained the same in some intervals in which the number of samples was increased, but the number of intensity levels actually decreased. The most likely reason for this result is that a decrease in $k$ tends to increase the apparent contrast, a visual effect often perceived as improved image quality.

## IMAGE INTERPOLATION

Interpolation is used in tasks such as zooming, shrinking, rotating, and geometrically correcting digital images. Our principal objective in this section is to introduce interpolation and apply it to image resizing (shrinking and zooming), which are basically image resampling methods. Uses of interpolation in applications such as rotation and geometric corrections will be discussed in Section 2.6.

*Interpolation* is the process of using known data to estimate values at unknown locations. We begin the discussion of this topic with a short example. Suppose that an image of size $500 \times 500$ pixels has to be enlarged 1.5 times to $750 \times 750$ pixels. A simple way to visualize zooming is to create an imaginary $750 \times 750$ grid with the same pixel spacing as the original image, then shrink it so that it exactly overlays the original image. Obviously, the pixel spacing in the shrunken $750 \times 750$ grid will be less than the pixel spacing in the original image. To assign an intensity value to any point in the overlay, we look for its closest pixel in the underlying original image and assign the intensity of that pixel to the new pixel in the $750 \times 750$ grid. When intensities have been assigned to all the points in the overlay grid, we expand it back to the specified size to obtain the resized image.

The method just discussed is called *nearest neighbor interpolation* because it assigns to each new location the intensity of its nearest neighbor in the original image (see Section 2.5 regarding neighborhoods). This approach is simple but, it has the tendency to produce undesirable artifacts, such as severe distortion of straight edges. A more suitable approach is *bilinear interpolation*, in which we use the four nearest neighbors to estimate the intensity at a given location. Let $(x, y)$ denote the coordinates of the location to which we want to assign an intensity value (think of it as a point of the grid described previously), and let $v(x, y)$ denote that intensity value. For bilinear interpolation, the assigned value is obtained using the equation

Contrary to what the name suggests, bilinear interpolation is *not* a linear operation because it involves multiplication of coordinates (which is not a linear operation). See Eq. (2-17).

$$v(x, y) = ax + by + cxy + d \tag{2-17}$$

where the four coefficients are determined from the four equations in four unknowns that can be written using the *four* nearest neighbors of point $(x, y)$. Bilinear interpolation gives much better results than nearest neighbor interpolation, with a modest increase in computational burden.

The next level of complexity is *bicubic interpolation*, which involves the sixteen nearest neighbors of a point. The intensity value assigned to point $(x, y)$ is obtained using the equation

$$v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \tag{2-18}$$

The sixteen coefficients are determined from the sixteen equations with sixteen unknowns that can be written using the sixteen nearest neighbors of point $(x, y)$. Observe that Eq. (2-18) reduces in form to Eq. (2-17) if the limits of both summations in the former equation are 0 to 1. Generally, bicubic interpolation does a better job of preserving fine detail than its bilinear counterpart. Bicubic interpolation is the standard used in commercial image editing applications, such as Adobe Photoshop and Corel Photopaint.

Although images are displayed with integer coordinates, it is possible during processing to work with *subpixel accuracy* by increasing the size of the image using interpolation to "fill the gaps" between pixels in the original image.

**EXAMPLE 2.4: Comparison of interpolation approaches for image shrinking and zooming.**

Figure 2.27(a) is the same as Fig. 2.23(d), which was obtained by reducing the resolution of the 930 dpi image in Fig. 2.23(a) to 72 dpi (the size shrank from $2136 \times 2140$ to $165 \times 166$ pixels) and then zooming the reduced image back to its original size. To generate Fig. 2.23(d) we used nearest neighbor interpolation both to shrink and zoom the image. As noted earlier, the result in Fig. 2.27(a) is rather poor. Figures 2.27(b) and (c) are the results of repeating the same procedure but using, respectively, bilinear and bicubic interpolation for both shrinking and zooming. The result obtained by using bilinear interpolation is a significant improvement over nearest neighbor interpolation, but the resulting image is blurred slightly. Much sharper results can be obtained using bicubic interpolation, as Fig. 2.27(c) shows.

a b c

**FIGURE 2.27** (a) Image reduced to 72 dpi and zoomed back to its original 930 dpi using nearest neighbor interpolation. This figure is the same as Fig. 2.23(d). (b) Image reduced to 72 dpi and zoomed using bilinear interpolation. (c) Same as (b) but using bicubic interpolation.

It is possible to use more neighbors in interpolation, and there are more complex techniques, such as using *splines* or *wavelets*, that in some instances can yield better results than the methods just discussed. While preserving fine detail is an exceptionally important consideration in image generation for 3-D graphics (for example, see Hughes and Andries [2013]), the extra computational burden seldom is justifiable for general-purpose digital image processing, where bilinear or bicubic interpolation typically are the methods of choice.

## 2.5 SOME BASIC RELATIONSHIPS BETWEEN PIXELS

In this section, we discuss several important relationships between pixels in a digital image. When referring in the following discussion to particular pixels, we use lowercase letters, such as $p$ and $q$.

### NEIGHBORS OF A PIXEL

A pixel $p$ at coordinates $(x, y)$ has two horizontal and two vertical neighbors with coordinates

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

This set of pixels, called the 4-*neighbors* of $p$, is denoted $N_4(p)$.

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

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

and are denoted $N_D(p)$. These neighbors, together with the 4-neighbors, are called the 8-*neighbors* of $p$, denoted by $N_8(p)$. The set of image locations of the neighbors of a point $p$ is called the *neighborhood* of $p$. The neighborhood is said to be *closed* if it contains $p$. Otherwise, the neighborhood is said to be *open*.

### ADJACENCY, CONNECTIVITY, REGIONS, AND BOUNDARIES

Let $V$ be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set $V$ typically contains more elements. For example, if we are dealing with the adjacency of pixels whose values are in the range 0 to 255, set $V$ could be any subset of these 256 values. We consider three types of adjacency:

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

**(a)** $q$ is in $N_4(p)$, *or*

**(b)** $q$ is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from $V$.

We use the symbols ∩ and ∪ to denote set intersection and union, respectively. Given sets $A$ and $B$, recall that their intersection is the set of elements that are members of both $A$ and $B$. The union of these two sets is the set of elements that are members of $A$, of $B$, or of both. We will discuss sets in more detail in Section 2.6.

Mixed adjacency is a modification of 8-adjacency, and is introduced to eliminate the ambiguities that may result from using 8-adjacency. For example, consider the pixel arrangement in Fig. 2.28(a) and let $V = \{1\}$. The three pixels at the top of Fig. 2.28(b) show multiple (ambiguous) 8-adjacency, as indicated by the dashed lines. This ambiguity is removed by using $m$-adjacency, as in Fig. 2.28(c). In other words, the center and upper-right diagonal pixels are not $m$-adjacent because they do not satisfy condition (b).

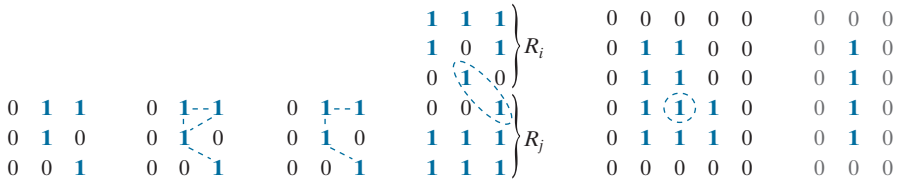
A *digital path* (or *curve*) from pixel $p$ with coordinates $(x_0, y_0)$ to pixel $q$ with coordinates $(x_n, y_n)$ is a sequence of distinct pixels with coordinates

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

where points $(x_i, y_i)$ and $(x_{i-1}, y_{i-1})$ are adjacent for $1 \le i \le n$. In this case, $n$ is the *length* of the path. If $(x_0, y_0) = (x_n, y_n)$ the path is a *closed* path. We can define 4-, 8-, or $m$-paths, depending on the type of adjacency specified. For example, the paths in Fig. 2.28(b) between the top right and bottom right points are 8-paths, and the path in Fig. 2.28(c) is an $m$-path.

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

Let $R$ represent a subset of pixels in an image. We call $R$ a *region* of the image if $R$ is a connected set. Two regions, $R_i$ and $R_j$ are said to be *adjacent* if their union forms a connected set. Regions that are not adjacent are said to be *disjoint*. We consider 4- and 8-adjacency when referring to regions. For our definition to make sense, the type of adjacency used must be specified. For example, the two regions of 1's in Fig. 2.28(d) are adjacent only if 8-adjacency is used (according to the definition in the previous




a b c d e f

**FIGURE 2.28** (a) An arrangement of pixels. (b) Pixels that are 8-adjacent (adjacency is shown by dashed lines). (c) $m$-adjacency. (d) Two regions (of 1's) that are 8-adjacent. (e) The circled point is on the boundary of the 1-valued pixels only if 8-adjacency between the region and background is used. (f) The inner boundary of the 1-valued region does not form a closed path, but its outer boundary does.

paragraph, a 4-path between the two regions does not exist, so their union is not a connected set).

Suppose an image contains $K$ disjoint regions, $R_k, k = 1, 2, \ldots, K$, none of which touches the image border.[†] Let $R_u$ denote the union of all the $K$ regions, and let $(R_u)^c$ denote its complement (recall that the *complement* of a set $A$ is the set of points that are not in $A$). We call all the points in $R_u$ the *foreground*, and all the points in $(R_u)^c$ the *background* of the image.

The *boundary* (also called the *border* or *contour*) of a region $R$ is the set of pixels in $R$ that are adjacent to pixels in the complement of $R$. Stated another way, the border of a region is the set of pixels in the region that have at least one background neighbor. Here again, we must specify the connectivity being used to define adjacency. For example, the point circled in Fig. 2.28(e) is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background, because the only possible connection between that point and the background is diagonal. As a rule, adjacency between points in a region and its background is defined using 8-connectivity to handle situations such as this.

The preceding definition sometimes is referred to as the *inner border* of the region to distinguish it from its *outer border*, which is the corresponding border in the background. This distinction is important in the development of border-following algorithms. Such algorithms usually are formulated to follow the outer boundary in order to guarantee that the result will form a closed path. For instance, the inner border of the 1-valued region in Fig. 2.28(f) is the region itself. This border does not satisfy the definition of a closed path. On the other hand, the outer border of the region does form a closed path around the region.

If $R$ happens to be an entire image, then its *boundary* (or *border*) is defined as the set of pixels in the first and last rows and columns of the image. This extra definition is required because an image has no neighbors beyond its border. Normally, when we refer to a region, we are referring to a subset of an image, and any pixels in the boundary of the region that happen to coincide with the border of the image are included implicitly as part of the region boundary.

The concept of an *edge* is found frequently in discussions dealing with regions and boundaries. However, there is a key difference between these two concepts. The boundary of a finite region forms a closed path and is thus a "global" concept. As we will discuss in detail in Chapter 10, edges are formed from pixels with derivative values that exceed a preset threshold. Thus, an edge is a "local" concept that is based on a measure of intensity-level discontinuity at a point. It is possible to link edge points into edge segments, and sometimes these segments are linked in such a way that they correspond to boundaries, but this is not always the case. The one exception in which edges and boundaries correspond is in binary images. Depending on the type of connectivity and edge operators used (we will discuss these in Chapter 10), the edge extracted from a binary region will be the same as the region boundary. This is

[†] We make this assumption to avoid having to deal with special cases. This can be done without loss of generality because if one or more regions touch the border of an image, we can simply pad the image with a 1-pixel-wide border of background values.

intuitive. Conceptually, until we arrive at Chapter 10, it is helpful to think of edges as intensity discontinuities, and of boundaries as closed paths.

## DISTANCE MEASURES

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

**(a)** $D(p,q) \geq 0 \quad (D(p,q) = 0 \text{ iff } p = q)$,
**(b)** $D(p,q) = D(q,p)$, and
**(c)** $D(p,s) \leq D(p,q) + D(q,s)$.

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

$$D_e(p,q) = \left[(x-u)^2 + (y-v)^2\right]^{\frac{1}{2}} \tag{2-19}$$

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

$$D_4(p,q) = |x-u| + |y-v| \tag{2-20}$$

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

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

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

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

$$D_8(p,q) = \max(|x-u|, |y-v|) \tag{2-21}$$

In this case, the pixels with $D_8$ distance from $(x, y)$ less than or equal to some value $d$ form a square centered at $(x, y)$. For example, the pixels with $D_8$ distance $\leq 2$ 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 the pixel at $(x, y)$.

Note that the $D_4$ and $D_8$ distances between $p$ and $q$ are independent of any paths that might exist between these points because these distances involve only the coordinates of the points. In the case of $m$-adjacency, however, the $D_m$ distance between two points is defined as the shortest $m$-path between the points. In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors. For instance, consider the following arrangement of pixels and assume that $p$, $p_2$, and $p_4$ have a value of 1, and that $p_1$ and $p_3$ can be 0 or 1:

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

Suppose that we consider adjacency of pixels valued 1 (i.e., $V = \{1\}$). If $p_1$ and $p_3$ are 0, the length of the shortest $m$-path (the $D_m$ distance) between $p$ and $p_4$ is 2. If $p_1$ is 1, then $p_2$ and $p$ will no longer be $m$-adjacent (see the definition of $m$-adjacency given earlier) and the length of the shortest $m$-path becomes 3 (the path goes through the points $pp_1p_2p_4$). Similar comments apply if $p_3$ is 1 (and $p_1$ is 0); in this case, the length of the shortest $m$-path also is 3. Finally, if both $p_1$ and $p_3$ are 1, the length of the shortest $m$-path between $p$ and $p_4$ is 4. In this case, the path goes through the sequence of points $pp_1p_2p_3p_4$.

## 2.6 INTRODUCTION TO THE BASIC MATHEMATICAL TOOLS USED IN DIGITAL IMAGE PROCESSING

You may find it helpful to download and study the review material dealing with probability, vectors, linear algebra, and linear systems. The review is available in the Tutorials section of the book website.

This section has two principal objectives: (1) to introduce various mathematical tools we use throughout the book; and (2) to help you begin developing a "feel" for how these tools are used by applying them to a variety of basic image-processing tasks, some of which will be used numerous times in subsequent discussions.

### ELEMENTWISE VERSUS MATRIX OPERATIONS

An *elementwise operation* involving one or more images is carried out on a *pixel-by-pixel* basis. We mentioned earlier in this chapter that images can be viewed equivalently as matrices. In fact, as you will see later in this section, there are many situations in which operations between images are carried out using matrix theory. It is for this reason that a clear distinction must be made between elementwise and matrix operations. For example, consider the following $2 \times 2$ images (matrices):

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

The elementwise product of two matrices is also called the *Hadamard product* of the matrices.

The *elementwise product* (often denoted using the symbol $\odot$ or $\otimes$) of these two images is

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

The symbol $\ominus$ is often used to denote *elementwise division*.

That is, the elementwise product is obtained by multiplying pairs of *corresponding* pixels. On the other hand, the *matrix product* of the images is formed using the rules of matrix multiplication:

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

We assume elementwise operations throughout the book, unless stated otherwise. For example, when we refer to raising an image to a power, we mean that each individual pixel is raised to that power; when we refer to dividing an image by another, we mean that the division is between corresponding pixel pairs, and so on. The terms *elementwise addition* and *subtraction* of two images are redundant because these are elementwise operations by definition. However, you may see them used sometimes to clarify notational ambiguities.

## LINEAR VERSUS NONLINEAR OPERATIONS

One of the most important classifications of an image processing method is whether it is linear or nonlinear. Consider a general operator, $\mathcal{H}$, that produces an output image, $g(x,y)$, from a given input image, $f(x,y)$:

$$\mathcal{H}[f(x,y)] = g(x,y) \tag{2-22}$$

Given two arbitrary constants, $a$ and $b$, and two arbitrary images $f_1(x,y)$ and $f_2(x,y)$, $\mathcal{H}$ is said to be a *linear operator* if

$$\begin{aligned} \mathcal{H}[af_1(x,y) + bf_2(x,y)] &= a\mathcal{H}[f_1(x,y)] + b\mathcal{H}[f_2(x,y)] \\ &= ag_1(x,y) + bg_2(x,y) \end{aligned} \tag{2-23}$$

This equation indicates that the output of a linear operation applied to the sum of two inputs is the same as performing the operation individually on the inputs and then summing the results. In addition, the output of a linear operation on a constant multiplied by an input is the same as the output of the operation due to the original input multiplied by that constant. The first property is called the property of *additivity,* and the second is called the property of *homogeneity*. By definition, an operator that fails to satisfy Eq. (2-23) is said to be *nonlinear*.

As an example, suppose that $\mathcal{H}$ is the sum operator, $\Sigma$. The function performed by this operator is simply to sum its inputs. To test for linearity, we start with the left side of Eq. (2-23) and attempt to prove that it is equal to the right side:

These are image summations, not the sums of all the elements of an image.

$$\begin{aligned} \sum[af_1(x,y) + bf_2(x,y)] &= \sum af_1(x,y) + \sum bf_2(x,y) \\ &= a\sum f_1(x,y) + b\sum f_2(x,y) \\ &= ag_1(x,y) + bg_2(x,y) \end{aligned}$$

where the first step follows from the fact that summation is distributive. So, an expansion of the left side is equal to the right side of Eq. (2-23), and we conclude that the sum operator is linear.

On the other hand, suppose that we are working with the max operation, whose function is to find the maximum value of the pixels in an image. For our purposes here, the simplest way to prove that this operator is nonlinear is to find an example that fails the test in Eq. (2-23). Consider the following two images

$$f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \qquad \text{and} \qquad f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$$

and suppose that we let $a = 1$ and $b = -1$. To test for linearity, we again start with the left side of Eq. (2-23):

$$\max\left\{(1)\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} + (-1)\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = \max\left\{\begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix}\right\}$$
$$= -2$$

Working next with the right side, we obtain

$$(1)\max\left\{\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}\right\} + (-1)\max\left\{\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} = 3 + (-1)7 = -4$$

The left and right sides of Eq. (2-23) are not equal in this case, so we have proved that the max operator is nonlinear.

As you will see in the next three chapters, linear operations are exceptionally important because they encompass a large body of theoretical and practical results that are applicable to image processing. The scope of nonlinear operations is considerably more limited. However, you will encounter in the following chapters several nonlinear image processing operations whose performance far exceeds what is achievable by their linear counterparts.

## ARITHMETIC OPERATIONS

Arithmetic operations between two images $f(x, y)$ and $g(x, y)$ are denoted as

$$\begin{aligned} s(x, y) &= f(x, y) + g(x, y) \\ d(x, y) &= f(x, y) - g(x, y) \\ p(x, y) &= f(x, y) \times g(x, y) \\ v(x, y) &= f(x, y) \div g(x, y) \end{aligned} \tag{2-24}$$

These are elementwise operations which, as noted earlier in this section, means that they are performed between corresponding pixel pairs in $f$ and $g$ for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$. As usual, $M$ and $N$ are the row and column sizes of the images. Clearly, $s$, $d$, $p$, and $v$ are images of size $M \times N$ also. Note that image arithmetic in the manner just defined involves images of the same size. The following examples illustrate the important role of arithmetic operations in digital image processing.

**EXAMPLE 2.5: Using image addition (averaging) for noise reduction.**

Suppose that $g(x, y)$ is a corrupted image formed by the addition of noise, $\eta(x, y)$, to a *noiseless* image $f(x, y)$; that is,

$$g(x, y) = f(x, y) + \eta(x, y) \tag{2-25}$$

where the assumption is that at every pair of coordinates $(x, y)$ the noise is uncorrelated[†] and has zero average value. We assume also that the noise and image values are uncorrelated (this is a typical assumption for additive noise). The objective of the following procedure is to reduce the noise content of the output image by adding a set of noisy input images, $\{g_i(x, y)\}$. This is a technique used frequently for image enhancement.

If the noise satisfies the constraints just stated, it can be shown (Problem 2.26) that if an image $\bar{g}(x, y)$ is formed by averaging $K$ different noisy images,

$$\bar{g}(x, y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x, y) \tag{2-26}$$

then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y) \tag{2-27}$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \tag{2-28}$$

where $E\{\bar{g}(x, y)\}$ is the expected value of $\bar{g}(x, y)$, and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\eta(x,y)}$ are the variances of $\bar{g}(x, y)$ and $\eta(x, y)$, respectively, all at coordinates $(x, y)$. These variances are arrays of the same size as the input image, and there is a scalar variance value for each pixel location.

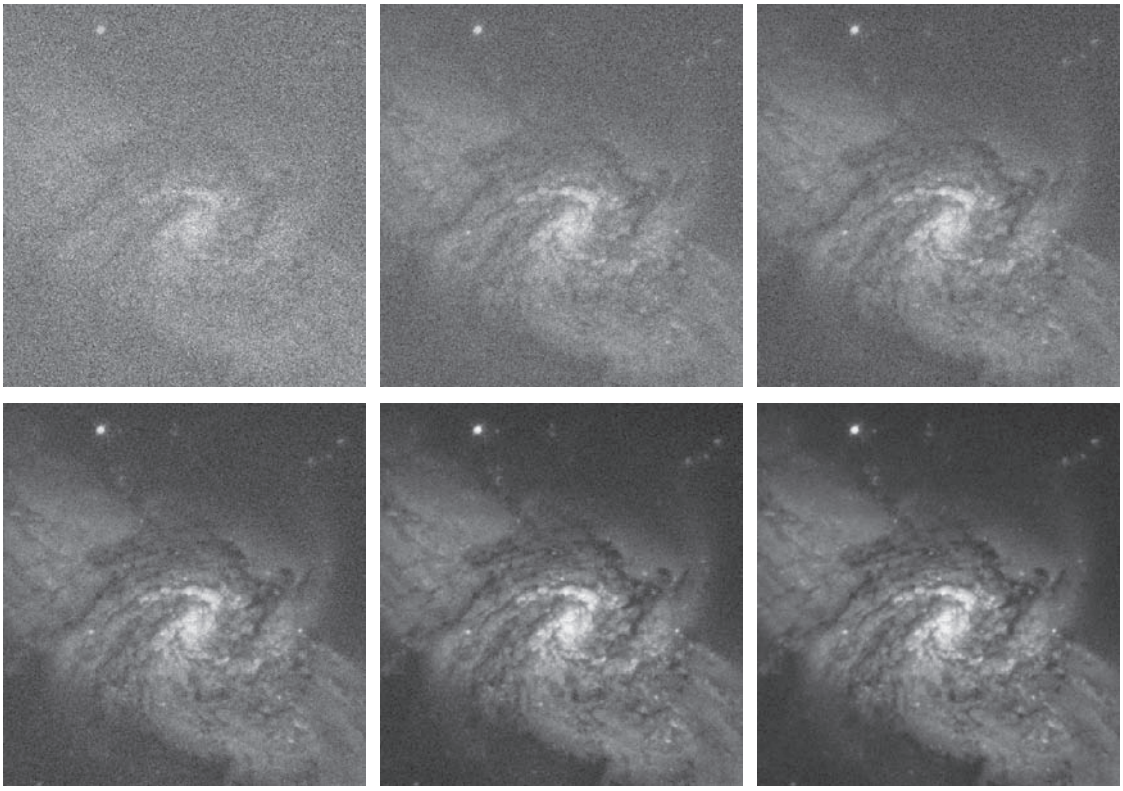
The standard deviation (square root of the variance) at any point $(x, y)$ in the average image is

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)} \tag{2-29}$$

As $K$ increases, Eqs. (2-28) and (2-29) indicate that the variability (as measured by the variance or the standard deviation) of the pixel values at each location $(x, y)$ decreases. Because $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ approaches the noiseless image $f(x, y)$ as the number of noisy images used in the averaging process increases. In order to avoid blurring and other artifacts in the output (average) image, it is necessary that the images $g_i(x, y)$ be *registered* (i.e., spatially aligned).

An important application of image averaging is in the field of astronomy, where imaging under very low light levels often cause sensor noise to render individual images virtually useless for analysis (lowering the temperature of the sensor helps reduce noise). Figure 2.29(a) shows an 8-bit image of the Galaxy Pair NGC 3314, in which noise corruption was simulated by adding to it Gaussian noise with zero mean and a standard deviation of 64 intensity levels. This image, which is representative of noisy astronomical images taken under low light conditions, is useless for all practical purposes. Figures 2.29(b) through (f) show the results of averaging 5, 10, 20, 50, and 100 images, respectively. We see from Fig. 2.29(b) that an average of only 10 images resulted in some visible improvement. According to Eq.

[†]The variance of a random variable $z$ with mean $\bar{z}$ is defined as $E\{(z - \bar{z})^2\}$, where $E\{\cdot\}$ is the expected value of the argument. The covariance of two random variables $z_i$ and $z_j$ is defined as $E\{(z_i - \bar{z}_i)(z_j - \bar{z}_j)\}$. If the variables are uncorrelated, their covariance is 0, and vice versa. (Do not confuse correlation and statistical independence. If two random variables are statistically independent, their correlation is zero. However, the converse is not true in general.)



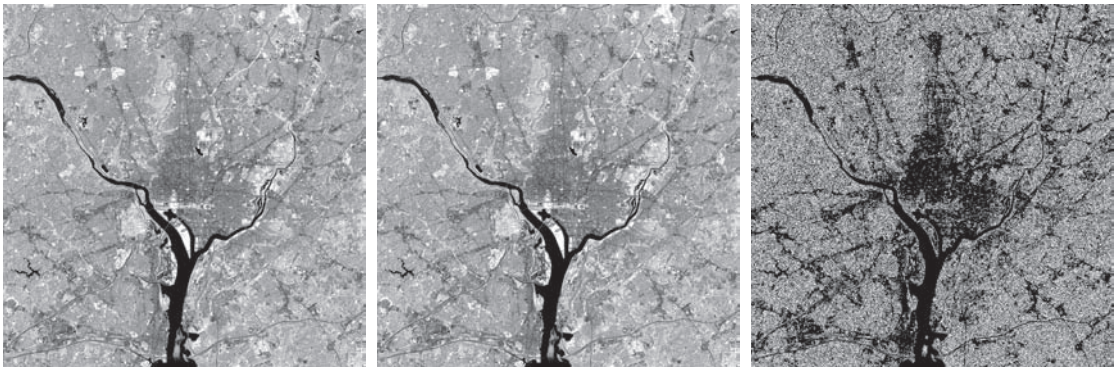
a b c
d e f

**FIGURE 2.29** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)-(f) Result of averaging 5, 10, 20, 50, and 1,00 noisy images, respectively. All images are of size $566 \times 598$ pixels, and all were scaled so that their intensities would span the full [0, 255] intensity scale. (Original image courtesy of NASA.)

(2-29), the standard deviation of the noise in Fig. 2.29(b) is less than half ($1/\sqrt{5} = 0.45$) the standard deviation of the noise in Fig. 2.29(a), or $(0.45)(64) \approx 29$ intensity levels. Similarly, the standard deviations of the noise in Figs. 2.29(c) through (f) are 0.32, 0.22, 0.14, and 0.10 of the original, which translates approximately into 20, 14, 9, and 6 intensity levels, respectively. We see in these images a progression of more visible detail as the standard deviation of the noise decreases. The last two images are visually identical for all practical purposes. This is not unexpected, as the difference between the standard deviations of their noise level is only about 3 intensity levels According to the discussion in connection with Fig. 2.5, this difference is below what a human generally is able to detect.

**EXAMPLE 2.6: Comparing images using subtraction.**

Image subtraction is used routinely for enhancing differences between images. For example, the image in Fig. 2.30(b) was obtained by setting to zero the least-significant bit of every pixel in Fig. 2.30(a). Visually, these images are indistinguishable. However, as Fig. 2.30(c) shows, subtracting one image from



a b c

**FIGURE 2.30** (a) Infrared image of the Washington, D.C. area. (b) Image resulting from setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity. (Original image courtesy of NASA.)

the other clearly shows their differences. Black (0) values in the difference image indicate locations where there is no difference between the images in Figs. 2.30(a) and (b).

We saw in Fig. 2.23 that detail was lost as the resolution was reduced in the chronometer image shown in Fig. 2.23(a). A vivid indication of image change as a function of resolution can be obtained by displaying the differences between the original image and its various lower-resolution counterparts. Figure 2.31(a) shows the difference between the 930 dpi and 72 dpi images. As you can see, the differences are quite noticeable. The intensity at any point in the difference image is proportional to the magnitude of the numerical difference between the two images at that point. Therefore, we can analyze which areas of the original image are affected the most when resolution is reduced. The next two images in Fig. 2.31 show proportionally less overall intensities, indicating smaller differences between the 930 dpi image and 150 dpi and 300 dpi images, as expected.

a b c

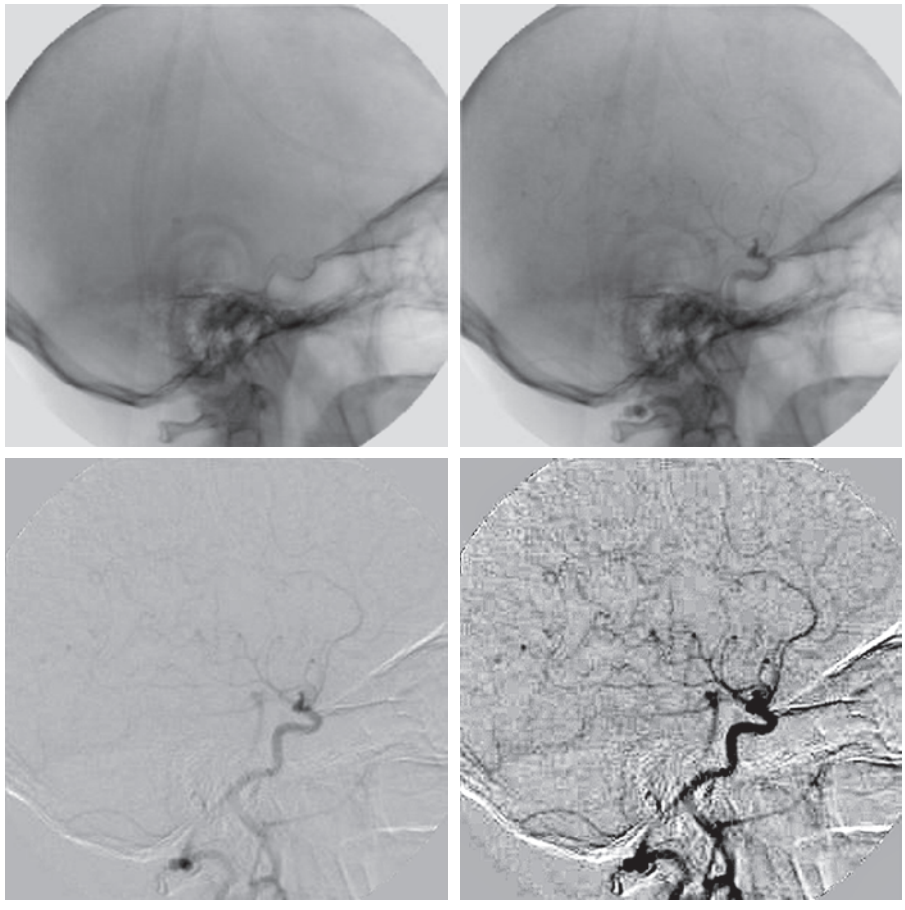
**FIGURE 2.31** (a) Difference between the 930 dpi and 72 dpi images in Fig. 2.23. (b) Difference between the 930 dpi and 150 dpi images. (c) Difference between the 930 dpi and 300 dpi images.

As a final illustration, we discuss briefly an area of medical imaging called *mask mode radiography*, a commercially successful and highly beneficial use of image subtraction. Consider image differences of the form

$$g(x, y) = f(x, y) - h(x, y) \tag{2-30}$$

In this case $h(x, y)$, the *mask*, is an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source. The procedure consists of injecting an X-ray contrast medium into the patient's bloodstream, taking a series of images called *live images* [samples of which are denoted as $f(x, y)$] of the same anatomical region as $h(x, y)$, and subtracting the mask from the series of incoming live images after injection of the contrast medium. The net effect of subtracting the mask from each sample live image is that the areas that are different between $f(x, y)$ and $h(x, y)$ appear in the output image, $g(x, y)$, as enhanced detail. Because images can be captured at TV rates, this procedure outputs a video showing how the contrast medium propagates through the various arteries in the area being observed.

Figure 2.32(a) shows a mask X-ray image of the top of a patient's head prior to injection of an iodine medium into the bloodstream, and Fig. 2.32(b) is a sample of a live image taken after the medium was



a b
c d

**FIGURE 2.32** Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of the Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

injected. Figure 2.32(c) is the difference between (a) and (b). Some fine blood vessel structures are visible in this image. The difference is clear in Fig. 2.32(d), which was obtained by sharpening the image and enhancing its contrast (we will discuss these techniques in the next chapter). Figure 2.32(d) is a "snapshot" of how the medium is propagating through the blood vessels in the subject's brain.

**EXAMPLE 2.7: Using image multiplication and division for shading correction and for masking.**

An important application of image multiplication (and division) is *shading correction*. Suppose that an imaging sensor produces images that can be modeled as the product of a "perfect image," denoted by $f(x, y)$, times a shading function, $h(x, y)$; that is, $g(x, y) = f(x, y)h(x, y)$. If $h(x, y)$ is known or can be estimated, we can obtain $f(x, y)$ (or an estimate of it) by multiplying the sensed image by the inverse of $h(x, y)$ (i.e., dividing $g$ by $h$ using elementwise division). If access to the imaging system is possible, we can obtain a good approximation to the shading function by imaging a target of constant intensity. When the sensor is not available, we often can estimate the shading pattern directly from a shaded image using the approaches discussed in Sections 3.5 and 9.8. Figure 2.33 shows an example of shading correction using an estimate of the shading pattern. The corrected image is not perfect because of errors in the shading pattern (this is typical), but the result definitely is an improvement over the shaded image in Fig. 2.33 (a). See Section 3.5 for a discussion of how we estimated Fig. 2.33 (b). Another use of image multiplication is in *masking*, also called *region of interest* (ROI), operations. As Fig. 2.34 shows, the process consists of multiplying a given image by a mask image that has 1's in the ROI and 0's elsewhere. There can be more than one ROI in the mask image, and the shape of the ROI can be arbitrary.

A few comments about implementing image arithmetic operations are in order before we leave this section. In practice, most images are displayed using 8 bits (even 24-bit color images consist of three separate 8-bit channels). Thus, we expect image values to be in the range from 0 to 255. When images are saved in a standard image format, such as TIFF or JPEG, conversion to this range is automatic. When image values exceed the allowed range, clipping or scaling becomes necessary. For example, the values in the difference of two 8-bit images can range from a minimum of –255

a b c

**FIGURE 2.33** Shading correction. (a) Shaded test pattern. (b) Estimated shading pattern. (c) Product of (a) by the reciprocal of (b). (See Section 3.5 for a discussion of how (b) was estimated.)

a b c

**FIGURE 2.34** (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

to a maximum of 255, and the values of the sum of two such images can range from 0 to 510. When converting images to eight bits, many software applications simply set all negative values to 0 and set to 255 all values that exceed this limit. Given a digital image $g$ resulting from one or more arithmetic (or other) operations, an approach guaranteeing that the full range of a values is "captured" into a fixed number of bits is as follows. First, we perform the operation

$$g_m = g - \min(g) \tag{2-31}$$

These are elementwise subtraction and division.

which creates an image whose minimum value is 0. Then, we perform the operation

$$g_s = K\left[g_m / \max(g_m)\right] \tag{2-32}$$

which creates a scaled image, $g_s$, whose values are in the range $[0, K]$. When working with 8-bit images, setting $K = 255$ gives us a scaled image whose intensities span the full 8-bit scale from 0 to 255. Similar comments apply to 16-bit images or higher. This approach can be used for all arithmetic operations. When performing division, we have the extra requirement that a small number should be added to the pixels of the divisor image to avoid division by 0.

## SET AND LOGICAL OPERATIONS

In this section, we discuss the basics of set theory. We also introduce and illustrate some important set and logical operations.

### Basic Set Operations

A *set* is a collection of distinct objects. If $a$ is an *element* of set $A$, then we write

$$a \in A \tag{2-33}$$

Similarly, if $a$ is not an element of $A$ we write

$$a \notin A \tag{2-34}$$

The set with no elements is called the *null* or *empty set*, and is denoted by $\varnothing$.

A set is denoted by the contents of two braces: { • }. For example, the expression

$$C = \{c \mid c = -d,\ d \in D\}$$

means that $C$ is the set of elements, $c$, such that $c$ is formed by multiplying each of the elements of set $D$ by –1.

If every element of a set $A$ is also an element of a set $B$, then $A$ is said to be a *subset* of $B$, denoted as

$$A \subseteq B \tag{2-35}$$

The *union* of two sets $A$ and $B$, denoted as

$$C = A \cup B \tag{2-36}$$

is a set $C$ consisting of elements belonging *either* to $A$, to $B$, *or* to *both*. Similarly, the *intersection* of two sets $A$ and $B$, denoted by

$$D = A \cap B \tag{2-37}$$

is a set $D$ consisting of elements belonging to *both* $A$ and $B$. Sets $A$ and $B$ are said to be *disjoint* or *mutually exclusive* if they have no elements in common, in which case,

$$A \cap B = \varnothing \tag{2-38}$$

The *sample space*, $\Omega$, (also called the *set universe*) is the set of all possible set elements in a given application. By definition, these set elements are members of the sample space for that application. For example, if you are working with the set of real numbers, then the sample space is the real line, which contains all the real numbers. In image processing, we typically define $\Omega$ to be the rectangle containing all the pixels in an image.

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

$$A^c = \{w \mid w \notin A\} \tag{2-39}$$

The *difference* of two sets $A$ and $B$, denoted $A - B$, is defined as

$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c \tag{2-40}$$

This is the set of elements that belong to $A$, but not to $B$. We can define $A^c$ in terms of $\Omega$ and the set difference operation; that is, $A^c = \Omega - A$. Table 2.1 shows several important set properties and relationships.

Figure 2.35 shows diagrammatically (in so-called *Venn diagrams*) some of the set relationships in Table 2.1. The shaded areas in the various figures correspond to the set operation indicated above or below the figure. Figure 2.35(a) shows the sample set, $\Omega$. As no earlier, this is the set of all possible elements in a given application. Figure 2.35(b) shows that the complement of a set $A$ is the set of all elements in $\Omega$ that are not in $A$, which agrees with our earlier definition. Observe that Figs. 2.35(e) and (g) are identical, which proves the validity of Eq. (2-40) using Venn diagrams. This

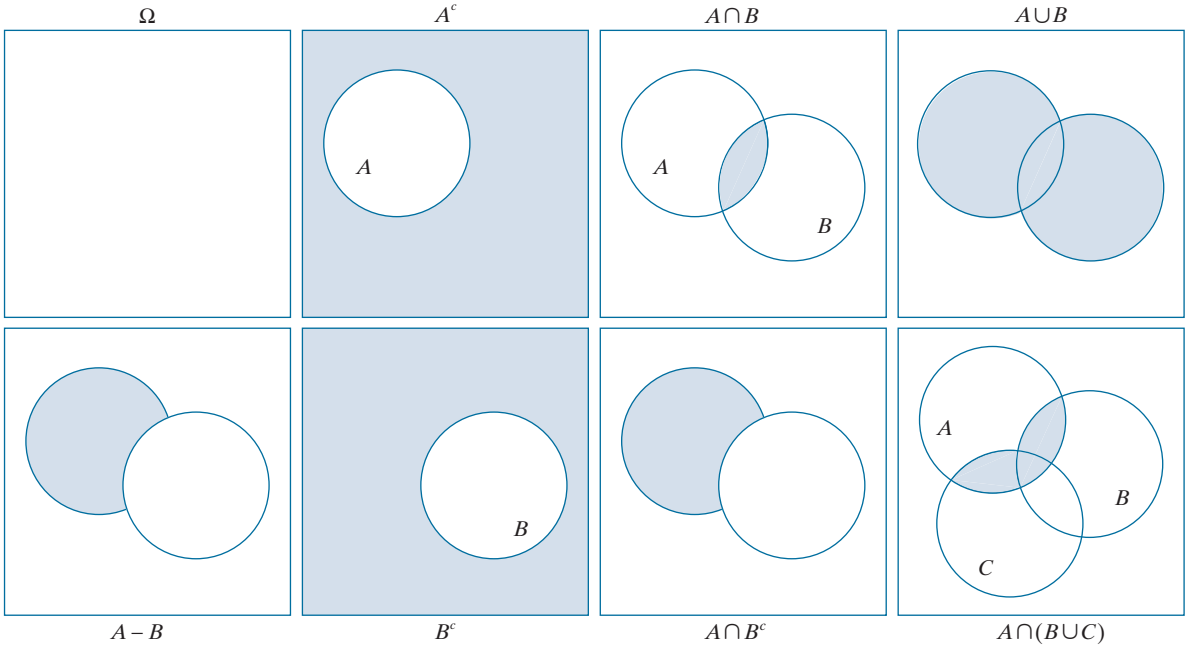
**TABLE 2.1**
Some important set operations and relationships.

| Description | Expressions |
|---|---|
| Operations between the sample space and null sets | $\Omega^c = \varnothing;\ \varnothing^c = \Omega;\ \Omega \cup \varnothing = \Omega;\ \Omega \cap \varnothing = \varnothing$ |
| Union and intersection with the null and sample space sets | $A \cup \varnothing = A;\ A \cap \varnothing = \varnothing;\ A \cup \Omega = \Omega;\ A \cap \Omega = A$ |
| Union and intersection of a set with itself | $A \cup A = A;\ A \cap A = A$ |
| Union and intersection of a set with its complement | $A \cup A^c = \Omega;\ A \cap A^c = \varnothing$ |
| Commutative laws | $A \cup B = B \cup A$<br>$A \cap B = B \cap A$ |
| Associative laws | $(A \cup B) \cup C = A \cup (B \cup C)$<br>$(A \cap B) \cap C = A \cap (B \cap C)$ |
| Distributive laws | $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$<br>$(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$ |
| DeMorgan's laws | $(A \cup B)^c = A^c \cap B^c$<br>$(A \cap B)^c = A^c \cup B^c$ |

is an example of the usefulness of Venn diagrams for proving equivalences between set relationships.

When applying the concepts just discussed to image processing, we let sets represent objects (regions) in a binary image, and the elements of the sets are the $(x, y)$ coordinates of those objects. For example, if we want to know whether two objects, $A$ and $B$, of a binary image overlap, all we have to do is compute $A \cap B$. If the result is not the empty set, we know that some of the elements of the two objects overlap. Keep in mind that the only way that the operations illustrated in Fig. 2.35 can make sense in the context of image processing is if the images containing the sets are binary, in which case we can talk about set membership based on coordinates, the assumption being that all members of the sets have the same intensity value (typically denoted by 1). We will discuss set operations involving binary images in more detail in the following section and in Chapter 9.

The preceding concepts are not applicable when dealing with grayscale images, because we have not defined yet a mechanism for assigning intensity values to the pixels resulting from a set operation. In Sections 3.8 and 9.6 we will define the union and intersection operations for grayscale values as the maximum and minimum of corresponding pixel pairs, respectively. We define the *complement* of a grayscale image as the pairwise differences between a constant and the intensity of every pixel in the image. The fact that we deal with corresponding pixel pairs tells us that grayscale set operations are elementwise operations, as defined earlier. The following example is a brief illustration of set operations involving grayscale images. We will discuss these concepts further in the two sections just mentioned.




a b c d
e f g h

**FIGURE 2.35** Venn diagrams corresponding to some of the set operations in Table 2.1. The results of the operations, such as $A^c$, are shown shaded. Figures (e) and (g) are the same, proving via Venn diagrams that $A - B = A \cap B^c$ [see Eq. (2-40)].

**EXAMPLE 2.8: Illustration of set operations involving grayscale images.**

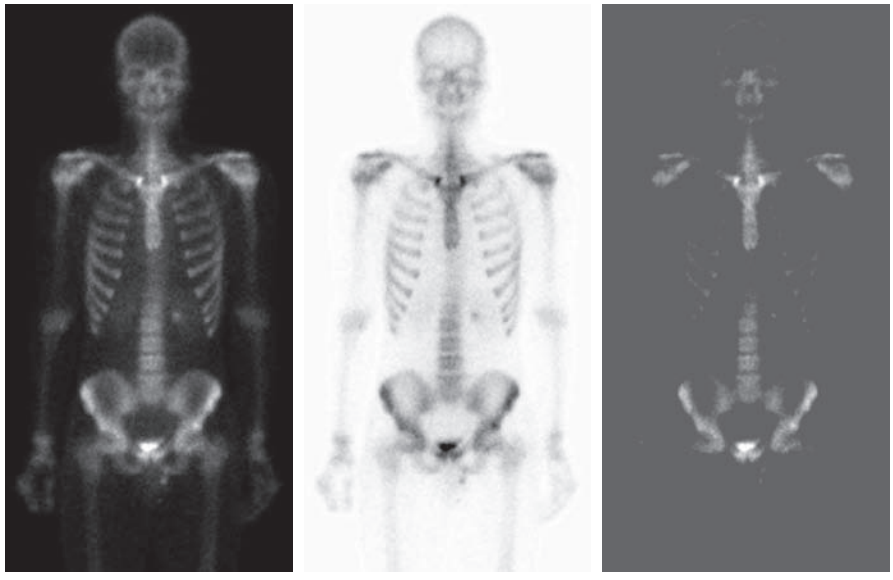
Let the elements of a grayscale image be represented by a set $A$ whose elements are triplets of the form $(x, y, z)$, where $x$ and $y$ are spatial coordinates, and $z$ denotes intensity values. We define the *complement* of $A$ as the set

$$A^c = \{(x, y, K - z) | (x, y, z) \in A\}$$

which is the set of pixels of $A$ whose intensities have been subtracted from a constant $K$. This constant is equal to the maximum intensity value in the image, $2^k - 1$, where $k$ is the number of bits used to represent $z$. Let $A$ denote the 8-bit grayscale image in Fig. 2.36(a), and suppose that we want to form the negative of $A$ using grayscale set operations. The negative is the set complement, and this is an 8-bit image, so all we have to do is let $K = 255$ in the set defined above:

$$A^c = \{(x, y, 255 - z) | (x, y, z) \in A\}$$

Figure 2.36(b) shows the result. We show this only for illustrative purposes. Image negatives generally are computed using an intensity transformation function, as discussed later in this section.



a b c

**FIGURE 2.36**
Set operations involving grayscale images. (a) Original image. (b) Image negative obtained using grayscale set complementation. (c) The union of image (a) and a constant image. (Original image courtesy of G.E. Medical Systems.)

The *union* of two grayscale sets $A$ and $B$ with the same number of elements is defined as the set

$$A \cup B = \left\{ \max_z(a, b) \,\middle|\, a \in A, b \in B \right\}$$

where it is understood that the max operation is applied to pairs of corresponding elements. If $A$ and $B$ are grayscale images of the same size, we see that their the union is an array formed from the maximum intensity between pairs of spatially corresponding elements. As an illustration, suppose that $A$ again represents the image in Fig. 2.36(a), and let $B$ denote a rectangular array of the same size as $A$, but in which all values of $z$ are equal to 3 times the mean intensity, $\overline{z}$, of the elements of $A$. Figure 2.36(c) shows the result of performing the set union, in which all values exceeding $3\overline{z}$ appear as values from $A$ and all other pixels have value $3\overline{z}$, which is a mid-gray value.

We follow convention in using the symbol $\times$ to denote the Cartesian product. This is not to be confused with our use of the same symbol throughout the book to denote the size of an $M$-by-$N$ image (i.e., $M \times N$).

Before leaving the discussion of sets, we introduce some additional concepts that are used later in the book. The *Cartesian product* of two sets $X$ and $Y$, denoted $X \times Y$, is the set of *all* possible ordered pairs whose first component is a member of $X$ and whose second component is a member of $Y$. In other words,

$$X \times Y = \left\{ (x, y) \,\middle|\, x \in X \text{ and } y \in Y \right\} \tag{2-41}$$

For example, if $X$ is a set of $M$ equally spaced values on the $x$-axis and $Y$ is a set of $N$ equally spaced values on the $y$-axis, we see that the Cartesian product of these two sets define the coordinates of an $M$-by-$N$ rectangular array (i.e., the coordinates of an image). As another example, if $X$ and $Y$ denote the specific $x$- and $y$-coordinates of a group of 8-connected, 1-valued pixels in a binary image, then set $X \times Y$ represents the region (object) comprised of those pixels.

A *relation* (or, more precisely, a *binary relation*) on a set $A$ is a collection of ordered pairs of elements from $A$. That is, a binary relation is a subset of the Cartesian product $A \times A$. A binary relation between *two* sets, $A$ and $B$, is a subset of $A \times B$.

A *partial order* on a set $S$ is a relation $\mathcal{R}$ on $S$ such that $\mathcal{R}$ is:

**(a)** *reflexive:* for any $a \in S$, $a\mathcal{R}a$;

**(b)** *transitive:* for any $a, b, c \in S$, $a\mathcal{R}b$ and $b\mathcal{R}c$ implies that $a\mathcal{R}c$;

**(c)** *antisymmetric:* for any $a, b \in S$, $a\mathcal{R}b$ and $b\mathcal{R}a$ implies that $a = b$.

where, for example, $a\mathcal{R}b$ reads "$a$ is related to $b$." This means that $a$ and $b$ are in set $\mathcal{R}$, which itself is a subset of $S \times S$ according to the preceding definition of a relation. A set with a partial order is called a *partially ordered set*.

Let the symbol $\preceq$ denote an ordering relation. An expression of the form

$$a_1 \preceq a_2 \preceq a_3 \preceq \cdots \preceq a_n$$

reads: $a_1$ precedes $a_2$ or is the same as $a_2$, $a_2$ precedes $a_3$ or is the same as $a_3$, and so on. When working with numbers, the symbol $\preceq$ typically is replaced by more traditional symbols. For example, the set of real numbers ordered by the relation "less than or equal to" (denoted by $\leq$) is a partially ordered set (see Problem 2.33). Similarly, the set of natural numbers, paired with the relation "divisible by" (denoted by $\div$), is a partially ordered set.

Of more interest to us later in the book are strict orderings. A *strict ordering* on a set $S$ is a relation $\mathcal{R}$ on $S$, such that $\mathcal{R}$ is:

**(a)** *antireflexive:* for any $a \in S$, $\neg a\mathcal{R}a$;

**(b)** *transitive:* for any $a, b, c \in S$, $a\mathcal{R}b$ and $b\mathcal{R}c$ implies that $a\mathcal{R}c$.

where $\neg a\mathcal{R}a$ means that $a$ is *not related* to $a$. Let the symbol $\prec$ denote a strict ordering relation. An expression of the form

$$a_1 \prec a_2 \prec a_3 \prec \cdots \prec a_n$$

reads $a_1$ precedes $a_2$, $a_2$ precedes $a_3$, and so on. A set with a strict ordering is called a *strict-ordered set*.

As an example, consider the set composed of the English alphabet of lowercase letters, $S = \{a, b, c, \cdots, z\}$. Based on the preceding definition, the ordering

$$a \prec b \prec c \prec \cdots \prec z$$

is strict because no member of the set can precede itself (antireflexivity) and, for any three letters in $S$, if the first precedes the second, and the second precedes the third, then the first precedes the third (transitivity). Similarly, the set of integers paired with the relation "less than ($<$)" is a strict-ordered set.

## Logical Operations

Logical operations deal with TRUE (typically denoted by 1) and FALSE (typically denoted by 0) variables and expressions. For our purposes, this means binary images

composed of *foreground* (1-valued) pixels, and a *background* composed of 0-valued pixels.

We work with set and logical operators on binary images using one of two basic approaches: (1) we can use the *coordinates* of individual regions of foreground pixels in a single image as sets, or (2) we can work with one or more images of the same size and perform logical operations between corresponding pixels in those arrays.

In the first category, a binary image can be viewed as a Venn diagram in which the coordinates of individual regions of 1-valued pixels are treated as sets. The union of these sets with the set composed of 0-valued pixels comprises the set universe, $\Omega$. In this representation, we work with single images using all the set operations defined in the previous section. For example, given a binary image with two 1-valued regions, $R_1$ and $R_2$, we can determine if the regions overlap (i.e., if they have at least one pair of coordinates in common) by performing the set intersection operation $R_1 \cap R_2$ (see Fig. 2.35). In the second approach, we perform logical operations on the pixels of one binary image, or on the corresponding pixels of two or more binary images of the same size.

Logical operators can be defined in terms of truth tables, as Table 2.2 shows for two logical variables $a$ and $b$. The logical AND operation (also denoted $\wedge$) yields a 1 (TRUE) only when both *a and b* are 1. Otherwise, it yields 0 (FALSE). Similarly, the logical OR ($\vee$) yields 1 when both *a or b* or *both* are 1, and 0 otherwise. The NOT ($\sim$) operator is self explanatory. When applied to two binary images, AND and OR operate on pairs of corresponding pixels between the images. That is, they are elementwise operators (see the definition of elementwise operators given earlier in this chapter) in this context. The operators AND, OR, and NOT are *functionally complete*, in the sense that they can be used as the basis for constructing any other logical operator.

Figure 2.37 illustrates the logical operations defined in Table 2.2 using the second approach discussed above. The NOT of binary image $B_1$ is an array obtained by changing all 1-valued pixels to 0, and vice versa. The AND of $B_1$ and $B_2$ contains a 1 at all spatial locations where the corresponding elements of $B_1$ and $B_2$ are 1; the operation yields 0's elsewhere. Similarly, the OR of these two images is an array that contains a 1 in locations where the corresponding elements of $B_1$, *or* $B_2$, *or both*, are 1. The array contains 0's elsewhere. The result in the fourth row of Fig. 2.37 corresponds to the set of 1-valued pixels in $B_1$ but not in $B_2$. The last row in the figure is the XOR (exclusive OR) operation, which yields 1 in the locations where the corresponding elements of $B_1$ *or* $B_2$, (but *not both*) are 1. Note that the logical

**TABLE 2.2**
Truth table defining the logical operators AND($\wedge$), OR($\vee$), and NOT($\sim$).

| $a$ | $b$ | $a$ AND $b$ | $a$ OR $b$ | NOT($a$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

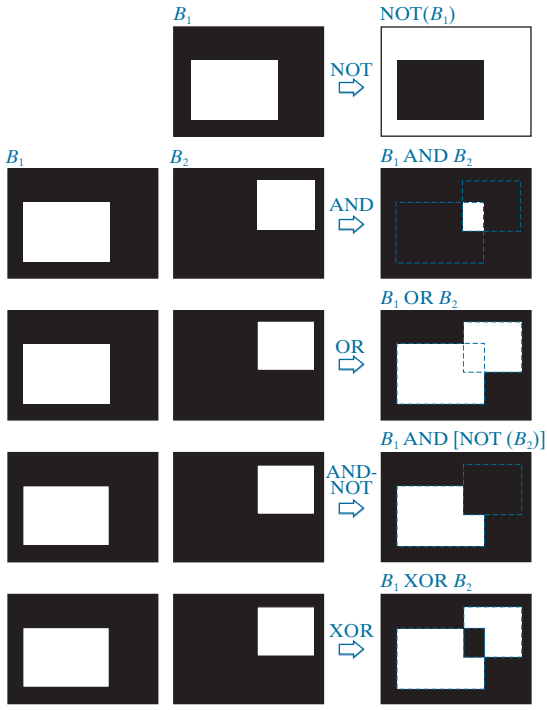



**FIGURE 2.37** Illustration of logical operations involving foreground (white) pixels. Black represents binary 0's and white binary 1's. The dashed lines are shown for reference only. They are not part of the result.

expressions in the last two rows of Fig. 2.37 were constructed using operators from Table 2.2; these are examples of the functionally complete nature of these operators.

We can arrive at the same results in Fig. 2.37 using the first approach discussed above. To do this, we begin by labeling the individual 1-valued regions in each of the two images (in this case there is only one such region in each image). Let $A$ and $B$ denote the *set of coordinates* of all the 1-valued pixels in images $B_1$ and $B_2$, respectively. Then we form a *single* array by ORing the two images, while keeping the labels $A$ and $B$. The result would look like the array $B_1$ OR $B_2$ in Fig. 2.37, but with the two white regions labeled $A$ and $B$. In other words, the resulting array would look like a Venn diagram. With reference to the Venn diagrams and set operations defined in the previous section, we obtain the results in the rightmost column of Fig. 2.37 using set operations as follows: $A^c = \text{NOT}(B_1)$, $A \cap B = B_1$ AND $B_2$, $A \cup B = B_1$ OR $B_2$, and similarly for the other results in Fig. 2.37. We will make extensive use in Chapter 9 of the concepts developed in this section.

## SPATIAL OPERATIONS

Spatial operations are performed directly on the pixels of an image. We classify spatial operations into three broad categories: (1) single-pixel operations, (2) neighborhood operations, and (3) geometric spatial transformations.

## Single-Pixel Operations

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

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

where $z$ is the intensity of a pixel in the original image and $s$ is the (mapped) intensity of the corresponding pixel in the processed image. For example, Fig. 2.38 shows the transformation used to obtain the *negative* (sometimes called the *complement*) of an 8-bit image. This transformation could be used, for example, to obtain the negative image in Fig. 2.36, instead of using sets.

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

## Neighborhood Operations

Let $S_{xy}$ denote the set of coordinates of a neighborhood (see Section 2.5 regarding neighborhoods) centered on an arbitrary point $(x, y)$ in an image, $f$. Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image, $g$, such that the value of that pixel is determined by a specified operation on the neighborhood of pixels in the input image with coordinates in the set $S_{xy}$. For example, suppose that the specified operation is to compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on $(x, y)$. The coordinates of pixels in this region are the elements of set $S_{xy}$. Figures 2.39(a) and (b) illustrate the process. We can express this averaging operation as

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

where $r$ and $c$ are the row and column coordinates of the pixels whose coordinates are in the set $S_{xy}$. Image $g$ is created by varying the coordinates $(x, y)$ so that the center of the neighborhood moves from pixel to pixel in image $f$, and then repeating the neighborhood operation at each new location. For instance, the image in Fig. 2.39(d) was created in this manner using a neighborhood of size $41 \times 41$. The

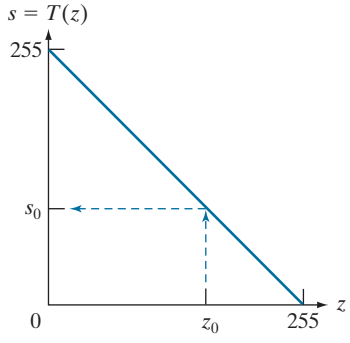
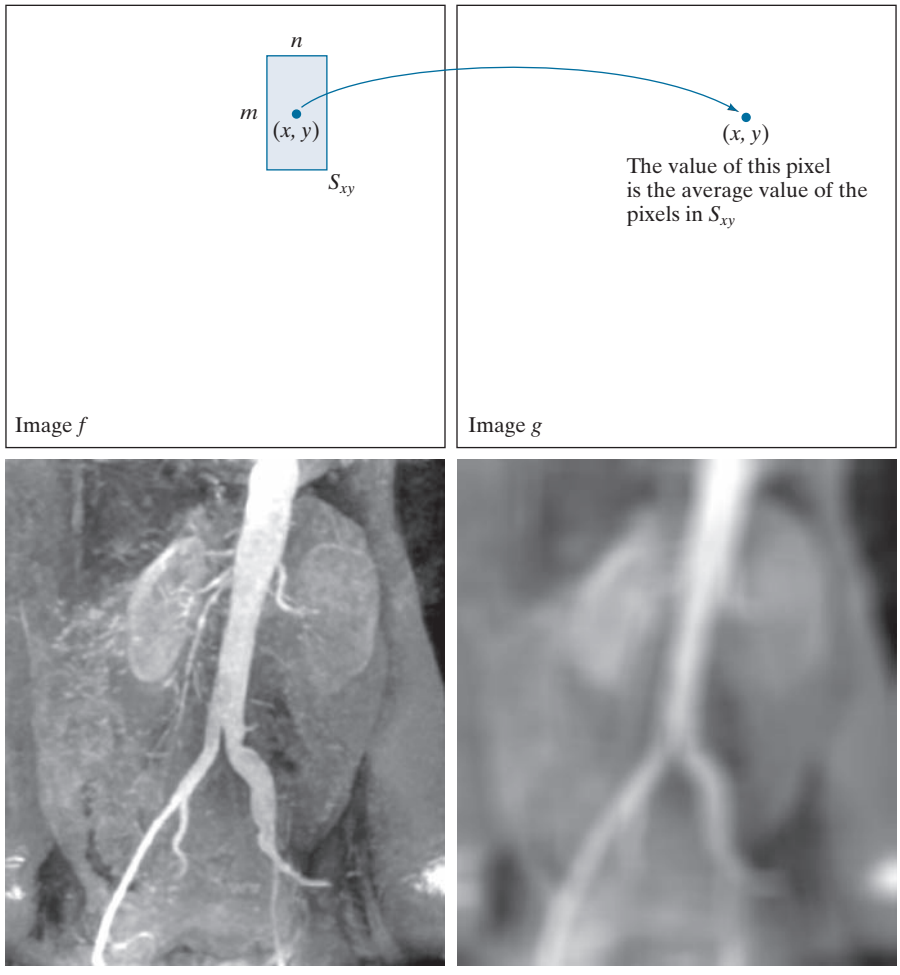



**FIGURE 2.38** Intensity transformation function used to obtain the digital equivalent of photographic negative of an 8-bit image..




a b
c d

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

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

## Geometric Transformations

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

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

The transformation of coordinates may be expressed as

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

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

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

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

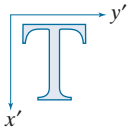
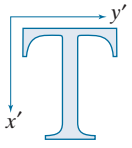
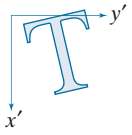
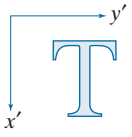
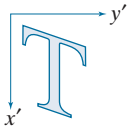
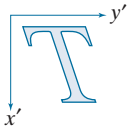
This transformation can *scale*, *rotate*, *translate*, or *sheer* an image, depending on the values chosen for the elements of matrix **A**. Table 2.3 shows the matrix values used to implement these transformations. A significant advantage of being able to perform all transformations using the unified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

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

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

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

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

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

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

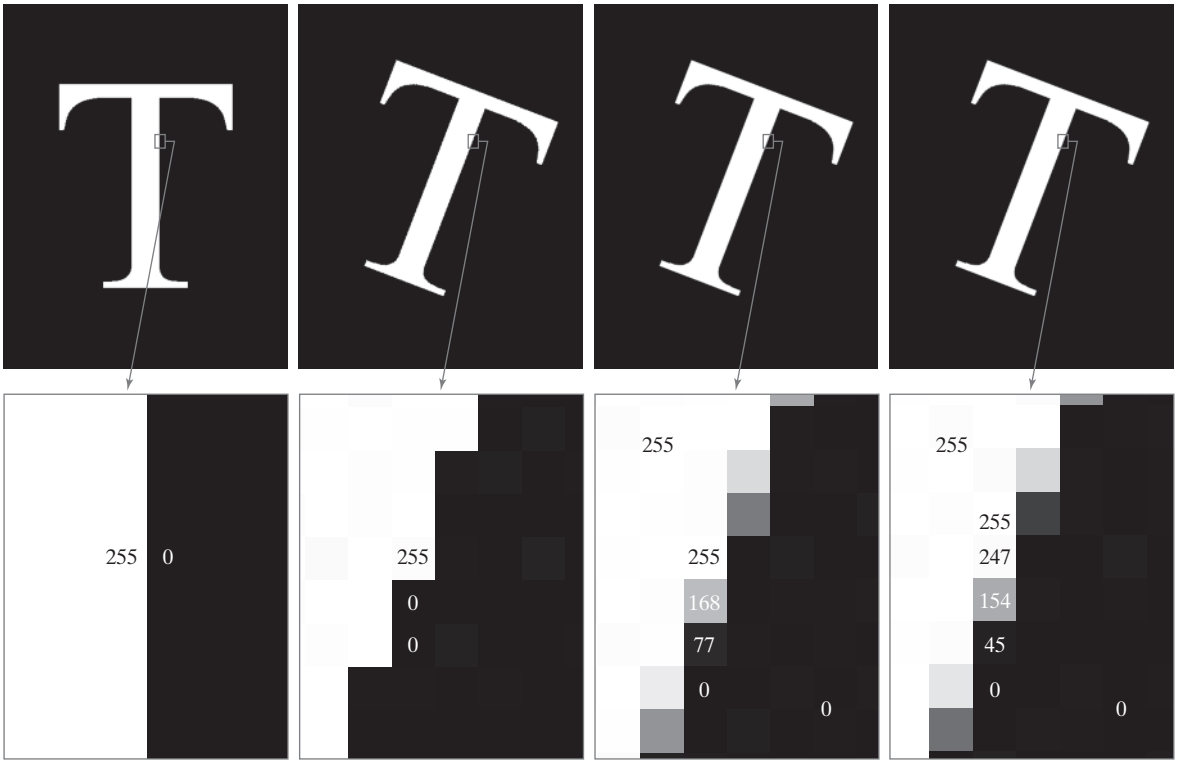
The objective of this example is to illustrate image rotation using an affine transform. Figure 2.40(a) shows a simple image and Figs. 2.40(b)–(d) are the results (using inverse mapping) of rotating the original image by –21° (in Table 2.3, clockwise angles of rotation are negative). Intensity assignments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

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

## Image Registration

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

Examples of image registration include aligning two or more images taken at approximately the same time, but using different imaging systems, such as an MRI (magnetic resonance imaging) scanner and a PET (positron emission tomography) scanner. Or, perhaps the images were taken at different times using the same instruments, such as satellite images of a given location taken several days, months, or even years apart. In either case, combining the images or performing quantitative analysis and comparisons between them requires compensating for geometric distortions caused by differences in viewing angle, distance, orientation, sensor resolution, shifts in object location, and other factors.




a b c d
e f g h

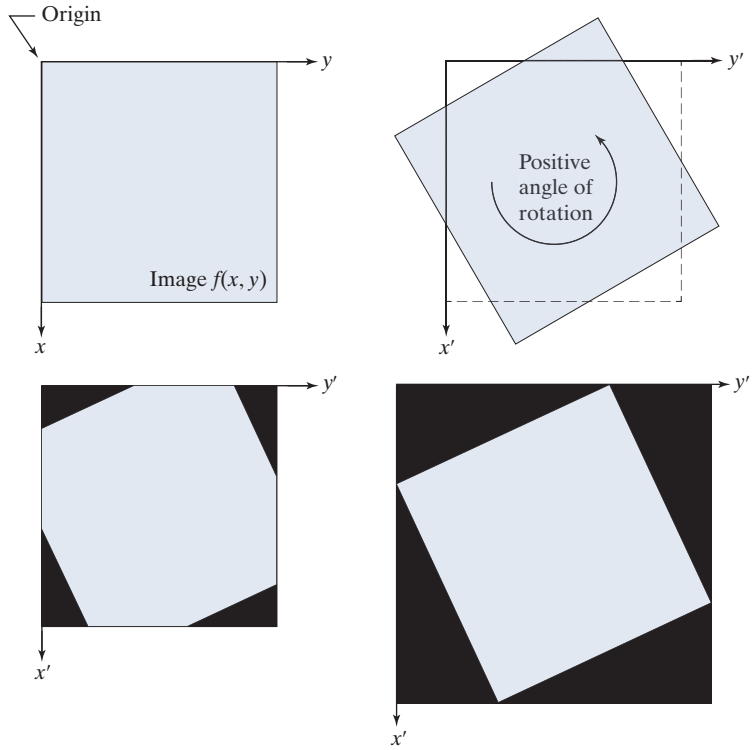
**FIGURE 2.40** (a) A $541 \times 421$ image of the letter T. (b) Image rotated –21° using nearest-neighbor interpolation for intensity assignments. (c) Image rotated –21° using bilinear interpolation. (d) Image rotated –21° using bicubic interpolation. (e)-(h) Zoomed sections (each square is one pixel,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

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

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

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

and




a b
c d

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

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

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

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

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

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

**EXAMPLE 2.10: Image registration.**

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

## VECTOR AND MATRIX OPERATIONS

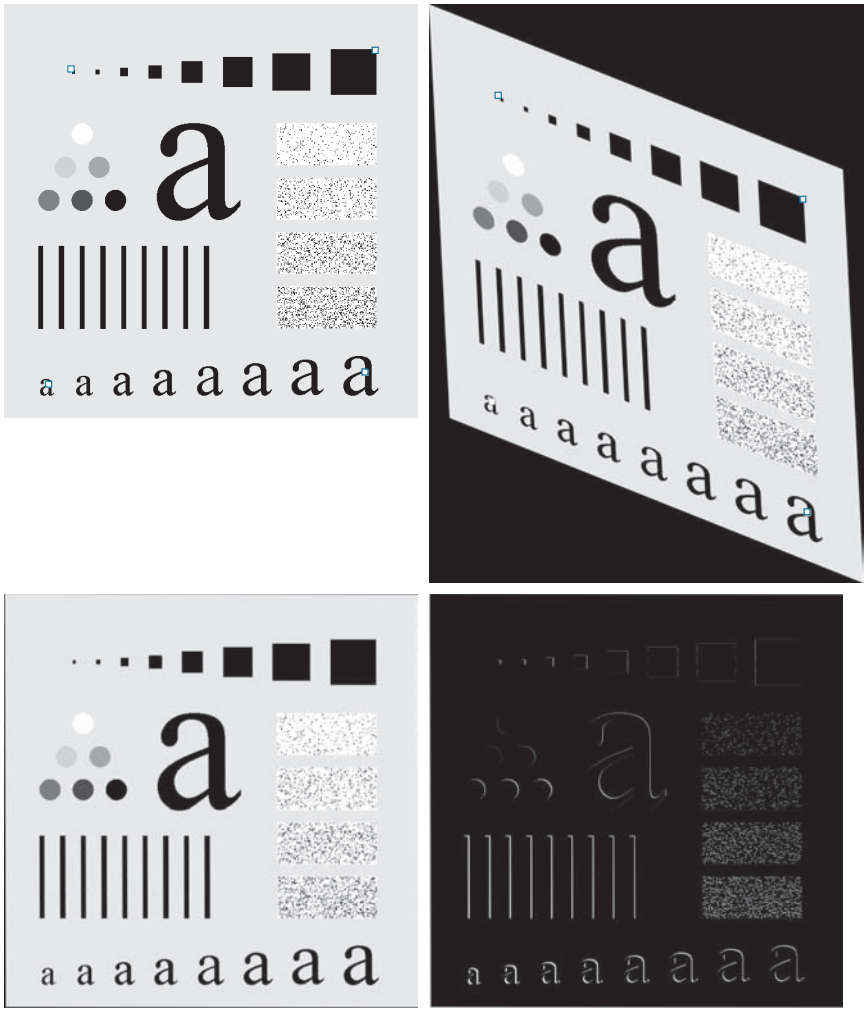
Multispectral image processing is a typical area in which vector and matrix operations are used routinely. For example, you will learn in Chapter 6 that color images are formed in RGB color space by using red, green, and blue component images, as Fig. 2.43 illustrates. Here we see that *each* pixel of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

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

where $z_1$ is the intensity of the pixel in the red image, and $z_2$ and $z_3$ are the corresponding pixel intensities in the green and blue images, respectively. Thus, an RGB color image of size $M \times N$ can be represented by three component images of this size, or by a total of $MN$ vectors of size $3 \times 1$. A general multispectral case involving $n$ component images (e.g., see Fig. 1.10) will result in $n$-dimensional vectors:

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

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



a b
c d

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

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

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

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

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

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

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

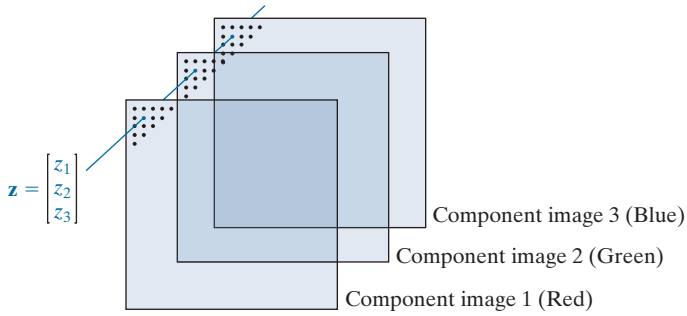



**FIGURE 2.43** Forming a vector from corresponding pixel values in three RGB component images.

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

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

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

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

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

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

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

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

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

where $\mathbf{f}$ is an $MN \times 1$ vector representing an input image, $\mathbf{n}$ is an $MN \times 1$ vector representing an $M \times N$ noise pattern, $\mathbf{g}$ is an $MN \times 1$ vector representing a processed image, and $\mathbf{H}$ is an $MN \times MN$ matrix representing a linear process applied to the input image (see the discussion earlier in this chapter regarding linear processes). It is possible, for example, to develop an entire body of generalized techniques for image restoration starting with Eq. (2-54), as we discuss in Section 5.9. We will mention the use of matrices again in the following section, and show other uses of matrices for image processing in numerous chapters in the book.

## IMAGE TRANSFORMS

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

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

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

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

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

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

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

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

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

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

Identical comments apply to the inverse kernel.

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

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

and

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

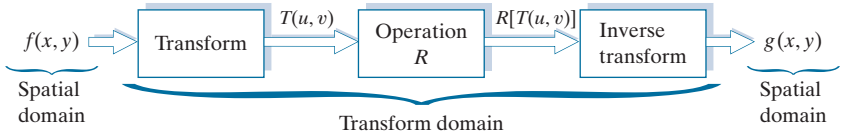



**FIGURE 2.44** General approach for working in the linear transform domain.

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

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

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

and

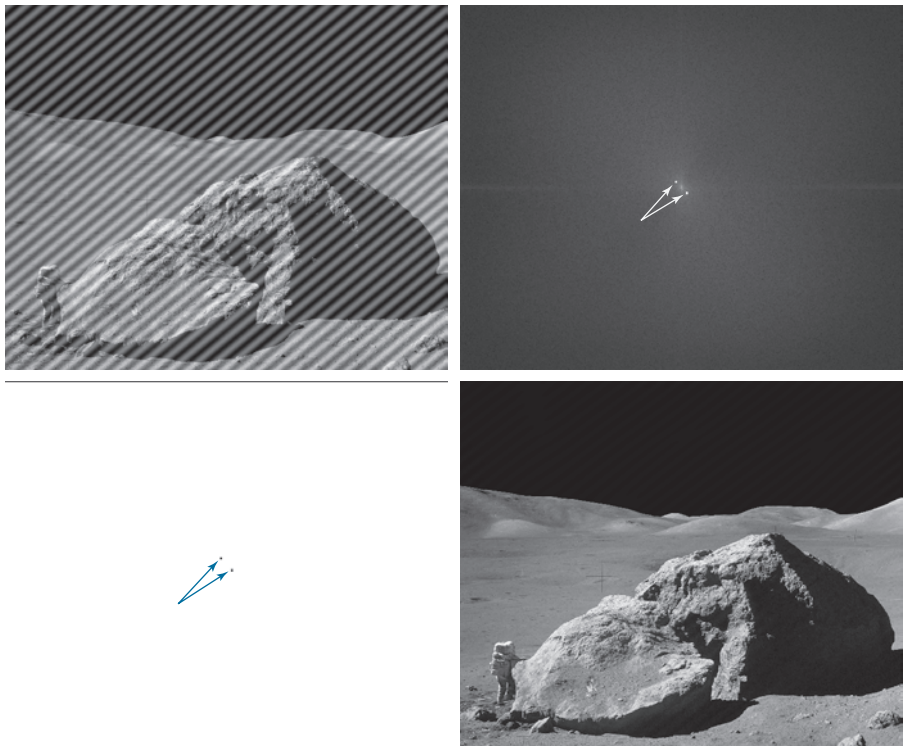
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)e^{j2\pi(ux/M + vy/N)} \tag{2-62}$$

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

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

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

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



a b
c d

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

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

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

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

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

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

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

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

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

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

## IMAGE INTENSITIES AS RANDOM VARIABLES

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

We treat image intensities as random quantities in numerous places in the book. For example, let $z_i, i = 0,1,2,\ldots,L-1$, denote the values of all possible intensities in an $M \times N$ digital image. The probability, $p(z_k)$, of intensity level $z_k$ occurring in the image is estimated as

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

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

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

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

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

Similarly, the variance of the intensities is

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

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

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

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

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

## Summary, References, and Further Reading

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

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

## Problems

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Start with this expression and

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

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

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

illuminated by a light source with intensity distribution

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

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

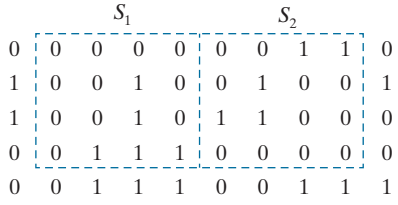
**2.13** Sketch the image in Problem 2.12 for $k = 2$.

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

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

**(b)** 8-adjacent.

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




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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**2.26** With reference to Example 2.5:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

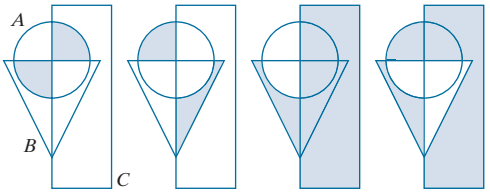
**(a)*** $(A \cap B) \cup [(A \cap C) - A \cap B \cap C] = A \cap (B \cup C)$

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

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

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

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




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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**2.39** Do the following.

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

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

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

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

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