6 Color Image Processing



> It is only after years of preparation that the young artist should touch color—not color used descriptively, that is, but as a means of personal expression.
>
> *Henri Matisse*
>
> For a long time I limited myself to one color—as a form of discipline.
>
> *Pablo Picasso*

Preview

Using color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades, compared to only about two dozen shades of gray. The latter factor is particularly important in manual image analysis. Color image processing is divided into two major areas: *pseudo-* and *full-color* processing. In the first category, the issue is one of assigning color(s) to a particular grayscale intensity or range of intensities. In the second, images typically are acquired using a full-color sensor, such as a digital camera, or color scanner. Until just a few years ago, most digital color image processing was done at the pseudo- or reduced-color level. However, because color sensors and processing hardware have become available at reasonable prices, full-color image processing techniques are now used in a broad range of applications. In the discussions that follow, it will become evident that some of the grayscale methods covered in previous chapters are applicable also to color images.

Upon completion of this chapter, readers should:

- Understand the fundamentals of color and the color spectrum.
- Be familiar with several of the color models used in digital image processing.
- Know how to apply basic techniques in pseudo-color image processing, including intensity slicing and intensity-to-color transformations.
- Be familiar with how to determine if a grayscale method is extendible to color images.
- Understand the basics of working with full-color images, including color transformations, color complements, and tone/color corrections.
- Be familiar with the role of noise in color image processing.
- Know how to perform spatial filtering on color images.
- Understand the advantages of using color in image segmentation.

6.1 COLOR FUNDAMENTALS

Although the process employed by the human brain in perceiving and interpreting color is a physiopsychological phenomenon that is not fully understood, the physical nature of color can be expressed on a formal basis supported by experimental and theoretical results.

In 1666, Sir Isaac Newton discovered that when a beam of sunlight passes through a glass prism, the emerging 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. 6.1 shows, the color spectrum may be divided into six broad regions: violet, blue, green, yellow, orange, and red. When viewed in full color (see Fig. 6.2), no color in the spectrum ends abruptly; rather, each color blends smoothly into the next.

Basically, the colors that humans and some other animals perceive in an object are determined by the nature of the light reflected from the object. As illustrated in Fig. 6.2, visible light is composed of a relatively narrow band of frequencies in the electromagnetic spectrum. A body that reflects light that is 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.

Characterization of light is central to the science of color. If the light is *achromatic* (void of color), its only attribute is its *intensity*, or amount. Achromatic light is what you see on movie films made before the 1930s. As defined in Chapter 2, and used numerous times since, the term *gray (*or *intensity) level* refers to a scalar measure of intensity that ranges from black, to grays, and finally to white.

Chromatic light spans the electromagnetic spectrum from approximately 400 to 700 nm. Three basic quantities used to describe the quality of a chromatic light source are: 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), is a measure of the amount of energy that 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, *brightness* is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of intensity, and is one of the key factors in describing color sensation.

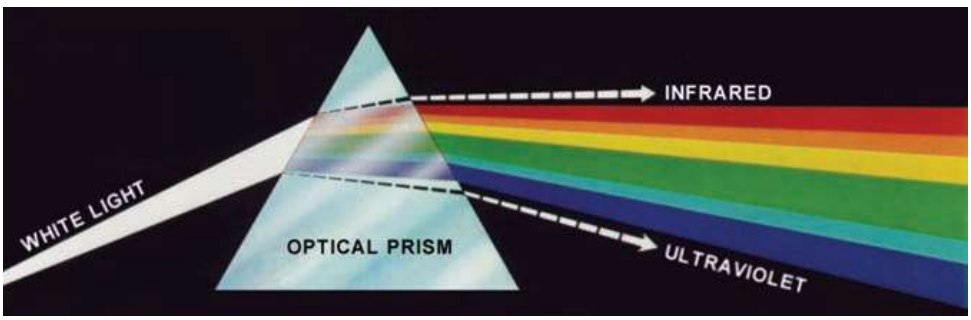


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lighting Division.)

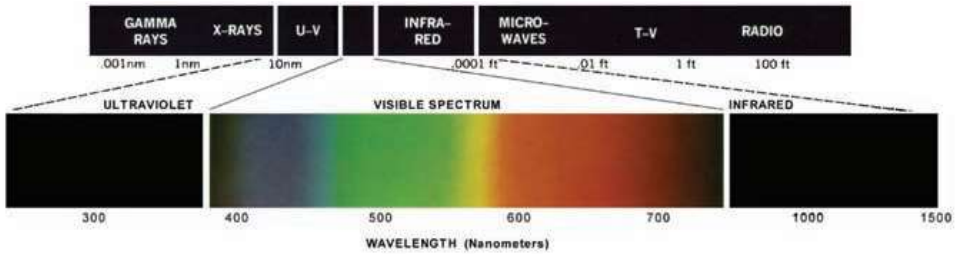


FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lighting Division.)

As noted in Section 2.1, cones are the sensors in the eye responsible for color vision. Detailed experimental evidence has established that the 6 to 7 million cones in the human eye can be divided into three principal sensing categories, corresponding roughly to red, green, and blue. Approximately 65% of all cones are sensitive to red light, 33% are sensitive to green light, and only about 2% are sensitive to blue. However, the blue cones are the most sensitive. Figure 6.3 shows average experimental curves detailing the absorption of light by the red, green, and blue cones in the eye. Because of these absorption characteristics, the human eye sees colors as variable combinations of the so-called *primary colors*: *red* (R), *green* (G), and *blue* (B).

For the purpose of standardization, the CIE (Commission Internationale de l'Eclairage—the International Commission on Illumination) designated in 1931 the following specific wavelength values to the three primary colors: blue = 435.8 nm, green = 546.1 nm, and red = 700 nm. This standard was set before results such as those in Fig. 6.3 became available in 1965. Thus, the CIE standards correspond only approximately with experimental data. It is important to keep in mind that defining three specific primary color wavelengths for the purpose of standardization does

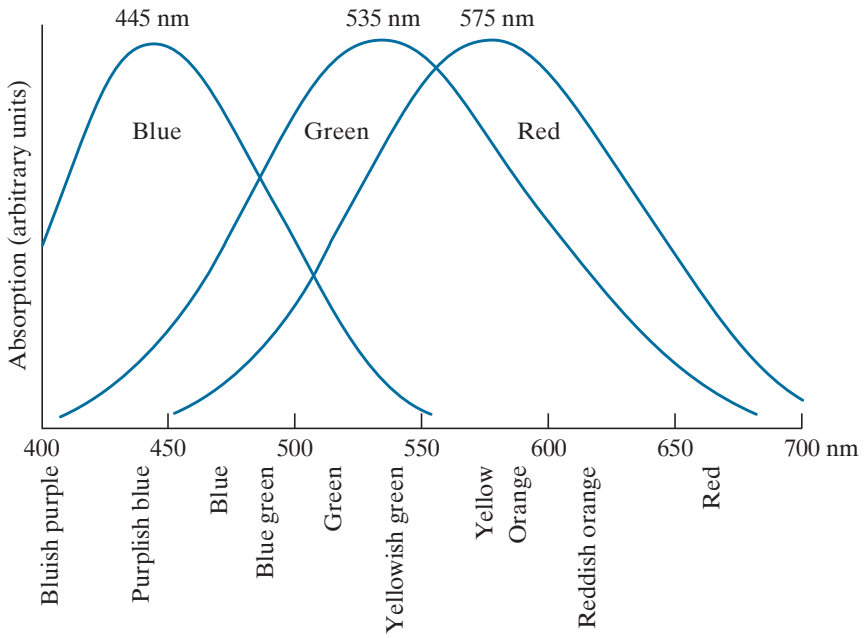


FIGURE 6.3 Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.

not mean that these three fixed RGB components acting alone can generate all spectrum colors. Use of the word *primary* has been widely misinterpreted to mean that the three standard primaries, when mixed in various intensity proportions, can produce all visible colors. As you will see shortly, this interpretation is not correct unless the wavelength also is allowed to vary, in which case we would no longer have three fixed primary colors.

The primary colors can be added together to produce the *secondary* colors of light—*magenta* (red plus blue), *cyan* (green plus blue), and *yellow* (red plus green). Mixing the three primaries, or a secondary with its opposite primary color, in the right intensities produces white light. This result is illustrated in Fig. 6.4(a), which shows also the three primary colors and their combinations to produce the secondary colors of light.

Differentiating between the primary colors of light and the primary colors of pigments or colorants is important. In the latter, a primary color is defined as one that subtracts or absorbs a primary color of light, and reflects or transmits the other two. Therefore, the primary colors of pigments are magenta, cyan, and yellow, and the secondary colors are red, green, and blue. These colors are shown in Fig. 6.4(b). A proper combination of the three pigment primaries, or a secondary with its opposite primary, produces black.

In practice, pigments seldom are pure. This results in a muddy brown instead of black when primaries, or primaries and secondaries, are combined. We will discuss this issue in Section 6.2

Color television reception is an example of the additive nature of light colors. The interior of CRT (cathode ray tube) color TV screens used well into the 1990s is composed of a large array of triangular dot patterns of electron-sensitive phosphor. When excited, each dot in a triad produces light in one of the primary colors. The

a
b

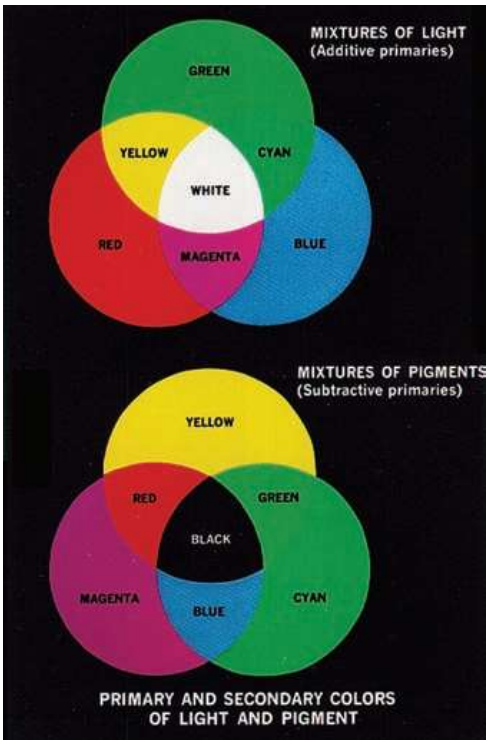


FIGURE 6.4 Primary and secondary colors of light and pigments. (Courtesy of the General Electric Co., Lighting Division.)

intensity of the red-emitting phosphor dots is modulated by an electron gun inside the tube, which generates pulses corresponding to the "red energy" seen by the TV camera. The green and blue phosphor dots in each triad are modulated in the same manner. The effect, viewed on the television receiver, is that the three primary colors from each phosphor triad are received and "added" together by the color-sensitive cones in the eye and perceived as a full-color image. Thirty successive image changes per second in all three colors complete the illusion of a continuous image display on the screen.

CRT displays started being replaced in the late 1990s by flat-panel digital technologies, such as liquid crystal displays (LCDs) and plasma devices. Although they are fundamentally different from CRTs, these and similar technologies use the same principle in the sense that they all require three subpixels (red, green, and blue) to generate a single color pixel. LCDs use properties of polarized light to block or pass light through the LCD screen and, in the case of active matrix display technologies, thin film transistors (TFTs) are used to provide the proper signals to address each pixel on the screen. Light filters are used to produce the three primary colors of light at each pixel triad location. In plasma units, pixels are tiny gas cells coated with phosphor to produce one of the three primary colors. The individual cells are addressed in a manner analogous to LCDs. This individual pixel triad coordinate addressing capability is the foundation of digital displays.

The characteristics generally used to distinguish one color from another are brightness, hue, and saturation. As indicated earlier in this section, brightness embodies the achromatic notion of intensity. *Hue* is an attribute associated with the dominant wavelength in a mixture of light waves. Hue represents dominant color as perceived by an observer. Thus, when we call an object red, orange, or yellow, we are referring to its hue. *Saturation* refers to the relative purity or the amount of white light mixed with a hue. The pure spectrum colors are fully saturated. Colors such as pink (red and white) and lavender (violet and white) are less saturated, with the degree of saturation being inversely proportional to the amount of white light added.

Hue and saturation taken together are called *chromaticity* and, therefore, a color may be characterized by its brightness and chromaticity. The amounts of red, green, and blue needed to form any particular color are called the *tristimulus* values, and are denoted, X, Y, and Z, respectively. A color is then specified by its *trichromatic coefficients*, defined as

$$x = \frac{X}{X + Y + Z} \tag{6-1}$$

$$y = \frac{Y}{X + Y + Z} \tag{6-2}$$

and

$$z = \frac{Z}{X + Y + Z} \tag{6-3}$$

Our use of x, y, and z in this context follows convention. These should not be confused with our use of (x, y) throughout the book to denote spatial coordinates.

We see from these equations that

$$x + y + z = 1 \tag{6-4}$$

For any wavelength of light in the visible spectrum, the tristimulus values needed to produce the color corresponding to that wavelength can be obtained directly from curves or tables that have been compiled from extensive experimental results (Poynton [1996, 2012]).

Another approach for specifying colors is to use the CIE *chromaticity diagram* (see Fig. 6.5), which shows color composition as a function of x (red) and y (green). For any value of x and y, the corresponding value of z (blue) is obtained from Eq. (6-4) by noting that $z = 1 - (x + y)$. The point marked green in Fig. 6.5, for example, has approximately 62% green and 25% red content. It follows from Eq. (6-4) that the composition of blue is approximately 13%.

The positions of the various spectrum colors—from violet at 380 nm to red at 780 nm—are indicated around the boundary of the tongue-shaped chromaticity diagram. These are the pure colors shown in the spectrum of Fig. 6.2. Any point not actually on the boundary, but within the diagram, represents some mixture of the pure spectrum colors. The *point of equal energy* shown in Fig. 6.5 corresponds to equal fractions of the three primary colors; it represents the CIE standard for white light. Any point located on the boundary of the chromaticity chart is fully saturated. As a point leaves the boundary and approaches the point of equal energy, more white light is added to the color, and it becomes less saturated. The saturation at the point of equal energy is zero.

The chromaticity diagram is useful for color mixing because a straight-line segment joining any two points in the diagram defines all the different color variations that can be obtained by combining these two colors additively. Consider, for example, a straight line drawn from the red to the green points shown in Fig. 6.5. If there is more red than green light, the exact point representing the new color will be on the line segment, but it will be closer to the red point than to the green point. Similarly, a line drawn from the point of equal energy to any point on the boundary of the chart will define all the shades of that particular spectrum color.

Extending this procedure to three colors is straightforward. To determine the range of colors that can be obtained from any three given colors in the chromaticity diagram, we simply draw connecting lines to each of the three color points. The result is a triangle, and any color inside the triangle, or on its boundary, can be produced by various combinations of the three vertex colors. A triangle with vertices at any three fixed colors cannot enclose the entire color region in Fig. 6.5. This observation supports graphically the remark made earlier that not all colors can be obtained with three single, *fixed* primaries, because three colors form a triangle.

The triangle in Fig. 6.6 shows a representative range of colors (called the *color gamut*) produced by RGB monitors. The shaded region inside the triangle illustrates the color gamut of today's high-quality color printing devices. The boundary of the color printing gamut is irregular because color printing is a combination of additive and subtractive color mixing, a process that is much more difficult to control than

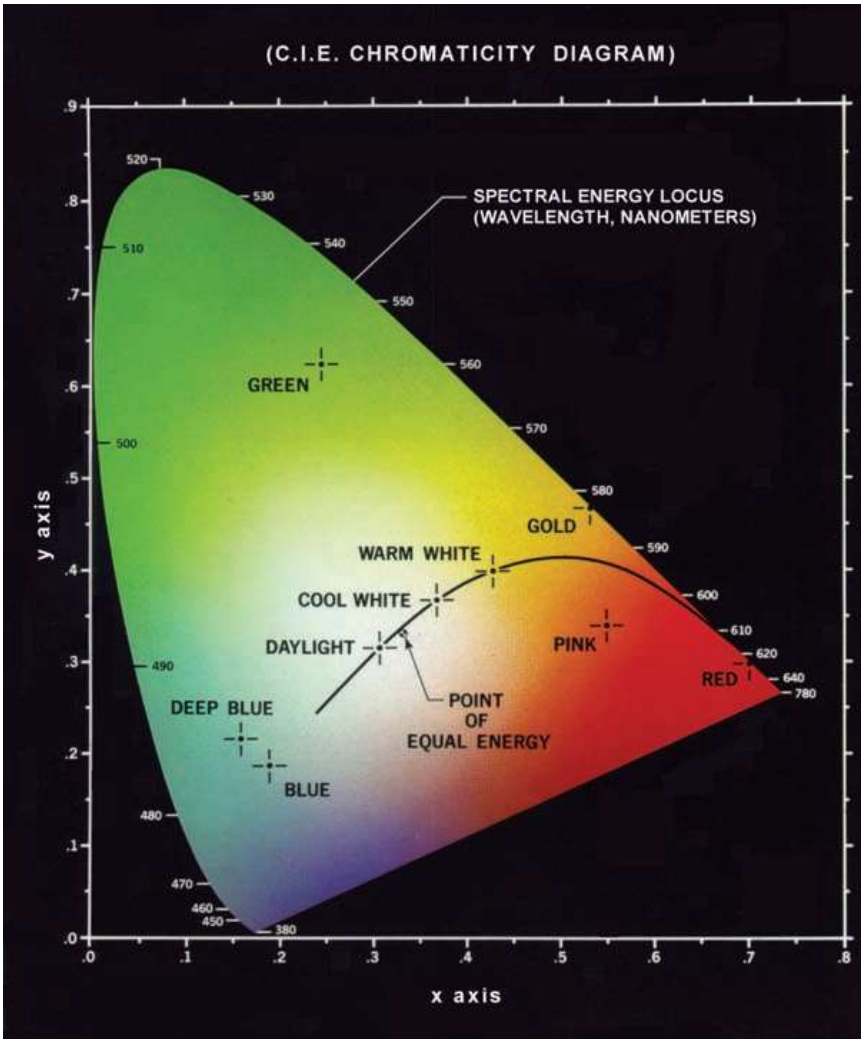


FIGURE 6.5
The CIE chromaticity diagram. (Courtesy of the General Electric Co., Lighting Division.)

that of displaying colors on a monitor, which is based on the addition of three highly controllable light primaries.

6.2 COLOR MODELS

The purpose of a *color model* (also called a *color space* or *color system*) is to facilitate the specification of colors in some standard way. In essence, a color model is a specification of (1) a coordinate system, and (2) a subspace within that system, such that each color in the model is represented by a single point contained in that subspace.

Most color models in use today are oriented either toward hardware (such as for color monitors and printers) or toward applications, where color manipulation is a goal (the creation of color graphics for animation is an example of the latter). In terms of digital image processing, the hardware-oriented models most commonly used in practice are the RGB (red, green, blue) model for color monitors and a

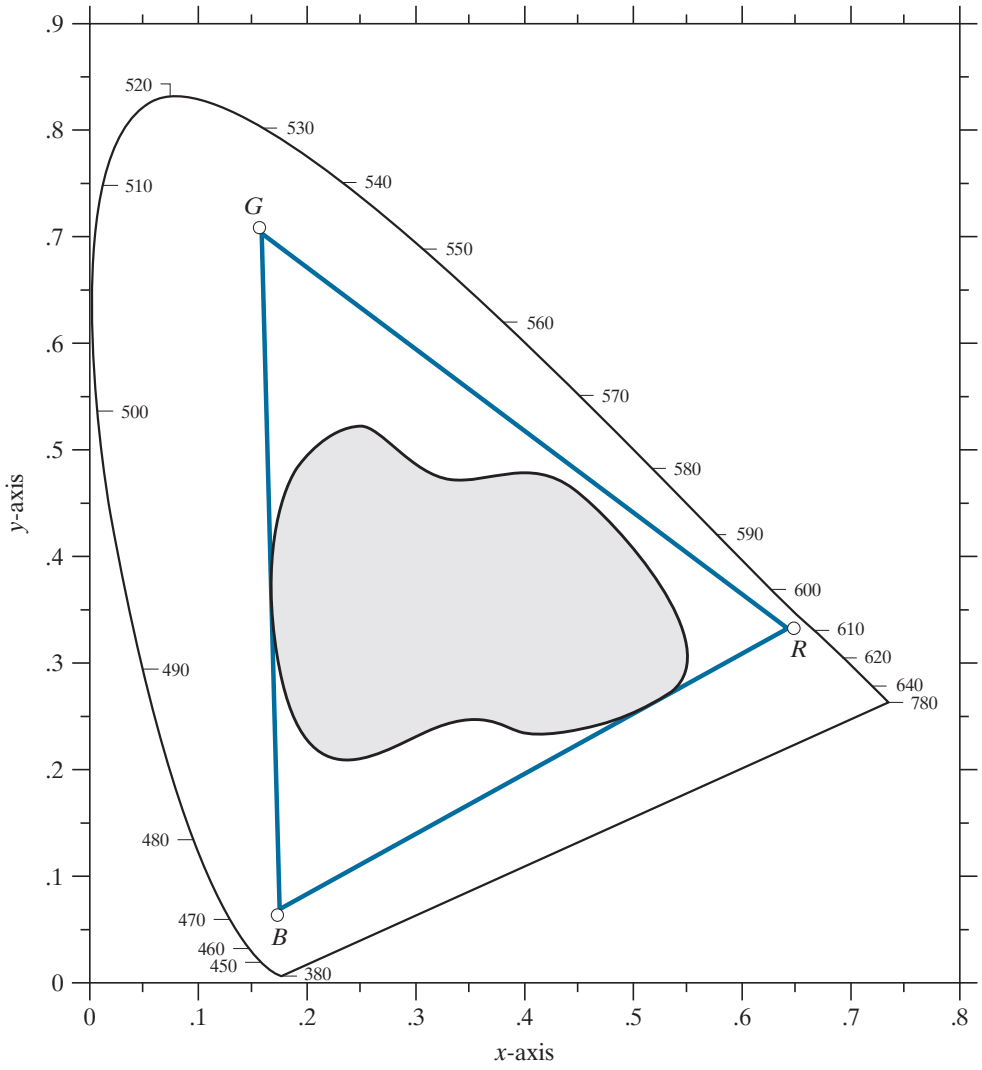


FIGURE 6.6 Illustrative color gamut of color monitors (triangle) and color printing devices (shaded region).

broad class of color video cameras; the CMY (cyan, magenta, yellow) and CMYK (cyan, magenta, yellow, black) models for color printing; and the HSI (hue, saturation, intensity) model, which corresponds closely with the way humans describe and interpret color. The HSI model also has the advantage that it decouples the color and gray-scale information in an image, making it suitable for many of the gray-scale techniques developed in this book. There are numerous color models in use today. This is a reflection of the fact that color science is a broad field that encompasses many areas of application. It is tempting to dwell on some of these models here, simply because they are interesting and useful. However, keeping to the task at hand, we focus attention on a few models that are representative of those used in image processing. Having mastered the material in this chapter, you will have no difficulty in understanding additional color models in use today.

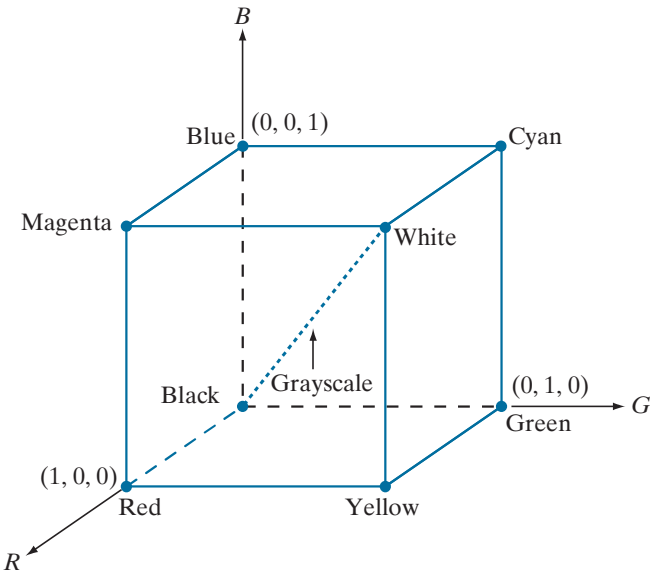


FIGURE 6.7 Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1, 1, 1).

THE RGB COLOR MODEL

In the RGB model, each color appears in its primary spectral components of red, green, and blue. This model is based on a Cartesian coordinate system. The color subspace of interest is the cube shown in Fig. 6.7, in which RGB primary values are at three corners; the secondary colors cyan, magenta, and yellow are at three other corners; black is at the origin; and white is at the corner farthest from the origin. In this model, the grayscale (points of equal RGB values) extends from black to white along the line joining these two points. The different colors in this model are points on or inside the cube, and are defined by vectors extending from the origin. For convenience, the assumption is that all color values have been normalized so the cube in Fig. 6.7 is the unit cube. That is, all values of R, G, and B in this representation are assumed to be in the range [0, 1]. Note that the RGB primaries can be interpreted as unit vectors emanating from the origin of the cube.

Images represented in the RGB color model consist of three component images, one for each primary color. When fed into an RGB monitor, these three images combine on the screen to produce a composite color image, as explained in Section 6.1. The number of bits used to represent each pixel in RGB space is called the *pixel depth*. Consider an RGB image in which each of the red, green, and blue images is an 8-bit image. Under these conditions, each RGB *color pixel* [that is, a triplet of values (R, G, B)] has a depth of 24 bits (3 image planes times the number of bits per plane). The term *full-color* image is used often to denote a 24-bit RGB color image. The total number of possible colors in a 24-bit RGB image is $(2^8)^3 = 16,777,216$. Figure 6.8 shows the 24-bit RGB color cube corresponding to the diagram in Fig. 6.7. Note also that for digital images, the range of values in the cube are scaled to the

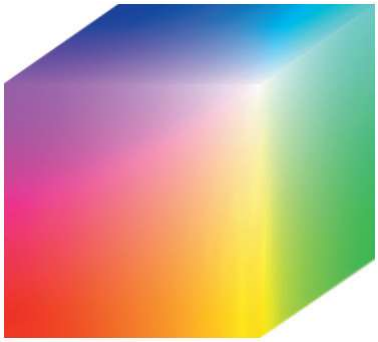


FIGURE 6.8
A 24-bit RGB color cube.

numbers representable by the number bits in the images. If, as above, the primary images are 8-bit images, the limits of the cube along each axis becomes [0, 255]. Then, for example, white would be at point [255, 255, 255] in the cube.

EXAMPLE 6.1: Generating a cross-section of the RGB color cube and its thee hidden planes.

The cube in Fig. 6.8 is a solid, composed of the $(2^8)^3$ colors mentioned in the preceding paragraph. A useful way to view these colors is to generate color planes (faces or cross sections of the cube). This is done by fixing one of the three colors and allowing the other two to vary. For instance, a cross-sectional plane through the center of the cube and parallel to the GB-plane in Fig. 6.8 is the plane (127, G, B) for $G, B = 0,1,2,\ldots,255$. Figure 6.9(a) shows that an image of this cross-sectional plane is generated by feeding the three individual component images into a color monitor. In the component images, 0 represents black and 255 represents white. Observe that each component image into the monitor is a grayscale image. The monitor does the job of combining the intensities of these images to generate an RGB image. Figure 6.9(b) shows the three hidden surface planes of the cube in Fig. 6.8, generated in a similar manner.

Acquiring a color image is the process shown in Fig. 6.9(a) in reverse. A color image can be acquired by using three filters, sensitive to red, green, and blue, respectively. When we view a color scene with a monochrome camera equipped with one of these filters, the result is a monochrome image whose intensity is proportional to the response of that filter. Repeating this process with each filter produces three monochrome images that are the RGB component images of the color scene. In practice, RGB color image sensors usually integrate this process into a single device. Clearly, displaying these three RGB component images as in Fig. 6.9(a) would yield an RGB color rendition of the original color scene.

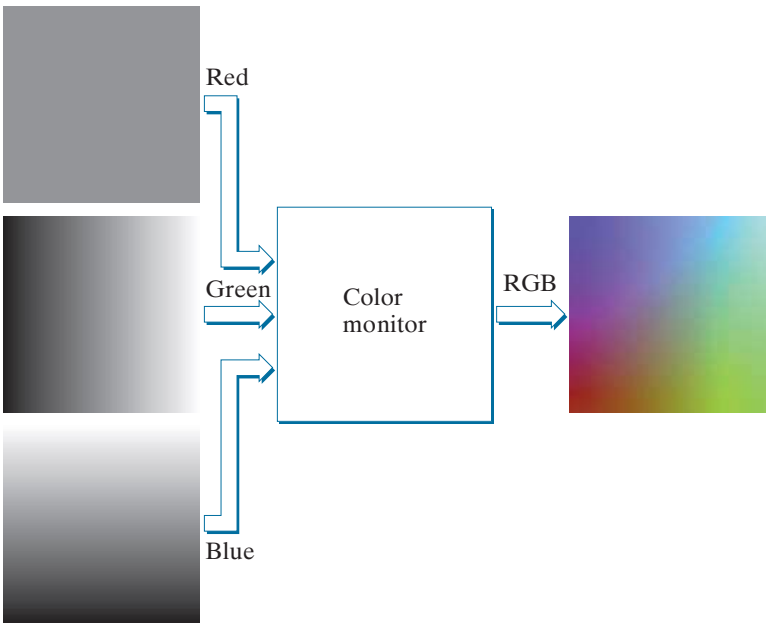
THE CMY AND CMYK COLOR MODELS

As indicated in Section 6.1, cyan, magenta, and yellow are the secondary colors of light or, alternatively, they are the primary colors of pigments. For example, when a surface coated with cyan pigment is illuminated with white light, no red light is reflected from the surface. That is, cyan subtracts red light from reflected white light, which itself is composed of equal amounts of red, green, and blue light.

Most devices that deposit colored pigments on paper, such as color printers and copiers, require CMY data input or perform an RGB to CMY conversion internally. This conversion is performed using the simple operation

a
b

FIGURE 6.9
(a) Generating the RGB image of the cross-sectional color plane (127, G, B). (b) The three hidden surface planes in the color cube of Fig. 6.8.



Equation (6-5), as well as all other equations in this section, are applied on a pixel-by-pixel basis.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6-5}$$

where the assumption is that all RGB color values have been normalized to the range [0, 1]. Equation (6-5) demonstrates that light reflected from a surface coated with pure cyan does not contain red (that is, $C = 1 - R$ in the equation). Similarly, pure magenta does not reflect green, and pure yellow does not reflect blue. Equation (6-5) also reveals that RGB values can be obtained easily from a set of CMY values by subtracting the individual CMY values from 1.

According to Fig. 6.4, equal amounts of the pigment primaries, cyan, magenta, and yellow, should produce black. In practice, because C, M, and Y inks seldom are pure colors, combining these colors for printing black produces instead a muddy-looking brown. So, in order to produce true black (which is the predominant color in printing), a fourth color, *black*, denoted by K, is added, giving rise to the CMYK color model. The black is added in just the proportions needed to produce true black. Thus,

when publishers talk about "four-color printing," they are referring to the three CMY colors, plus a portion of black.

The conversion from CMY to CMYK begins by letting

$$K = \min(C, M, Y) \tag{6-6}$$

If $K = 1$, then we have pure black, with no color contributions, from which it follows that

$$C = 0 \tag{6-7}$$

$$M = 0 \tag{6-8}$$

$$Y = 0 \tag{6-9}$$

The C, M, and Y on the right side of Eqs. (6-6)-(6-12) are in the CMY color system. The C, M, and Y on the left of Eqs. (6-7)-(6-12) are in the CMYK system.

Otherwise,

$$C = (C - K)/(1 - K) \tag{6-10}$$

$$M = (M - K)/(1 - K) \tag{6-11}$$

$$Y = (Y - K)/(1 - K) \tag{6-12}$$

where all values are assumed to be in the range [0, 1]. The conversions from CMYK back to CMY are:

The C, M, Y, and K on the right side of Eqs. (6-13)-(6-15) are in the CMYK color system. The C, M, and Y on the left of these equations are in the CMY system.

$$C = C * (1 - K) + K \tag{6-13}$$

$$M = M * (1 - K) + K \tag{6-14}$$

$$Y = Y * (1 - Y) + K \tag{6-15}$$

As noted at the beginning of this section, all operations in the preceding equations are performed on a pixel-by-pixel basis. Because we can use Eq. (6-5) to convert both ways between CMY and RGB, we can use that equation as a "bridge" to convert between RGB and CMYK, and vice versa.

It is important to keep in mind that all the conversions just presented to go between RGB, CMY, and CMYK are based on the preceding relationships as a group. There are many other ways to convert between these color models, so you cannot mix approaches and expect to get meaningful results. Also, colors seen on monitors generally appear much different when printed, unless these devices are calibrated (see the discussion of a device-independent color model later in this section). The same holds true in general for colors converted from one model to another. However, our interest in this chapter is not on color fidelity; rather, we are interested in using the properties of color models to facilitate image processing tasks, such as region detection.

THE HSI COLOR MODEL

As we have seen, creating colors in the RGB, CMY, and CMYK models, and changing from one model to the other, is straightforward. These color systems are ideally suited for hardware implementations. In addition, the RGB system matches nicely with the fact that the human eye is strongly perceptive to red, green, and blue primaries. Unfortunately, the RGB, CMY, and other similar color models are not well suited for describing colors in terms that are practical for human interpretation. For example, one does not refer to the color of an automobile by giving the percentage of each of the primaries composing its color. Furthermore, we do not think of color images as being composed of three primary images that combine to form a single image.

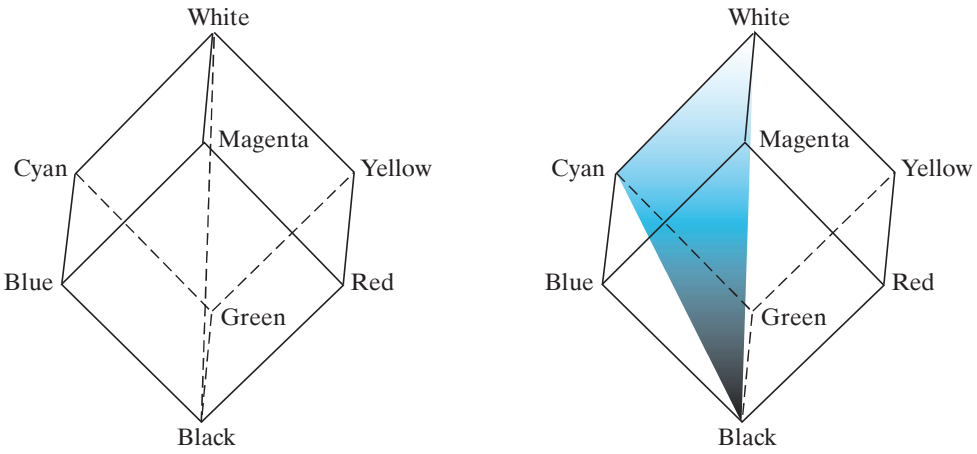
When humans view a color object, we describe it by its hue, saturation, and brightness. Recall from the discussion in Section 6.1 that hue is a color attribute that describes a pure color (pure yellow, orange, or red), whereas saturation gives a measure of the degree to which a pure color is diluted by white light. Brightness is a subjective descriptor that is practically impossible to measure. It embodies the achromatic notion of *intensity* and is one of the key factors in describing color sensation. We do know that intensity (gray level) is a most useful descriptor of achromatic images. This quantity definitely is measurable and easily interpretable. The model we are about to present, called the *HSI* (hue, saturation, intensity) *color model*, decouples the intensity component from the color-carrying information (hue and saturation) in a color image. As a result, the HSI model is a useful tool for developing image processing algorithms based on color descriptions that are natural and intuitive to humans, who, after all, are the developers and users of these algorithms. We can summarize by saying that RGB is ideal for image color generation (as in image capture by a color camera or image display on a monitor screen), but its use for color description is much more limited. The material that follows provides an effective way to do this.

We know from Example 6.1 that an RGB color image is composed three grayscale intensity images (representing red, green, and blue), so it should come as no surprise that we can to extract intensity from an RGB image. This becomes clear if we take the color cube from Fig. 6.7 and stand it on the black, (0, 0, 0), vertex, with the white, (1, 1, 1), vertex directly above it [see Fig. 6.10(a)]. As noted in our discussion of Fig. 6.7, the intensity (gray) scale is along the line joining these two vertices. In Figs. 6.10(a) and (b), the line (intensity axis) joining the black and white vertices is vertical. Thus, if we wanted to determine the intensity component of any color point in Fig. 6.10, we would simply define a plane that contains the color point and, at the same time, is perpendicular to the intensity axis. The intersection of the plane with the intensity axis would give us a point with intensity value in the range $[0, 1]$. A little thought would reveal that the saturation (purity) of a color increases as a function of distance from the intensity axis. In fact, the saturation of points on the intensity axis is zero, as evidenced by the fact that all points along this axis are gray.

Hue can be determined from an RGB value also. To see how, consider Fig. 6.10(b), which shows a plane defined by three points (black, white, and cyan). The fact that

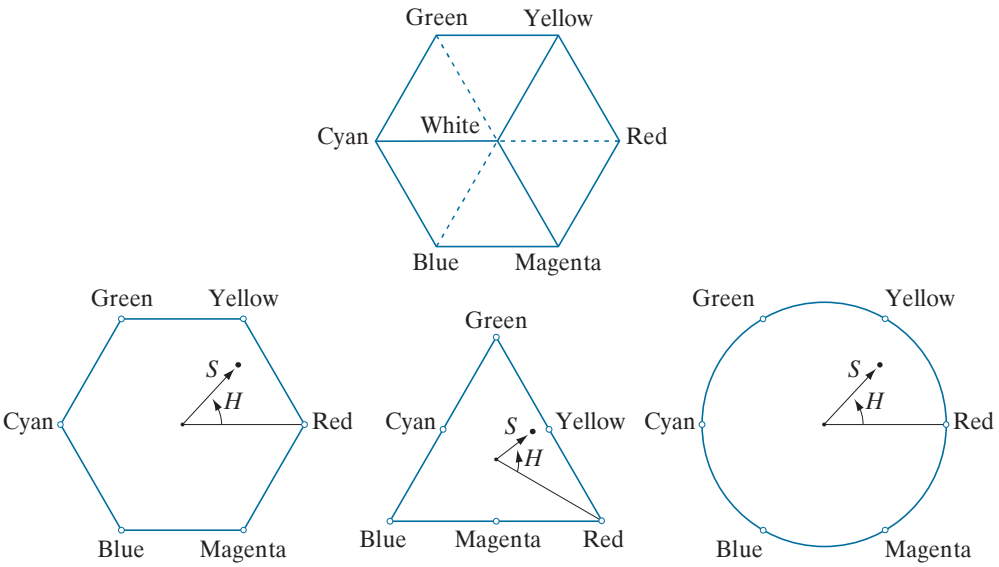
a b

FIGURE 6.10 Conceptual relationships between the RGB and HSI color models.



the black and white points are contained in the plane tells us that the intensity axis also is contained in the plane. Furthermore, we see that *all* points contained in the plane segment defined by the intensity axis and the boundaries of the cube have the *same* hue (cyan in this case). We could arrive at the same conclusion by recalling from Section 6.1 that all colors generated by three colors lie in the triangle defined by those colors. If two of those points are black and white, and the third is a color point, all points on the triangle would have the same hue, because the black and white components cannot change the hue (of course, the intensity and saturation of points in this triangle would be different). By rotating the shaded plane about the vertical intensity axis, we would obtain different hues. From these concepts, we arrive at the conclusion that the hue, saturation, and intensity values required to form the HSI space can be obtained from the RGB color cube. That is, we can convert any RGB point to a corresponding point in the HSI color space by working out the formulas that describe the reasoning outlined in the preceding discussion.

The key point regarding the cube arrangement in Fig. 6.10, and its corresponding HSI color space, is that the HSI space is represented by a vertical intensity axis, and the locus of color points that lie on planes perpendicular to that axis. As the planes move up and down the intensity axis, the boundaries defined by the intersection of each plane with the faces of the cube have either a triangular or a hexagonal shape. This can be visualized much more readily by looking at the cube straight down its grayscale axis, as shown in Fig. 6.11(a). We see that the primary colors are separated by 120°. The secondary colors are 60° from the primaries, which means that the angle between secondaries is 120° also. Figure 6.11(b) shows the same hexagonal shape and an arbitrary color point (shown as a dot). The hue of the point is determined by an angle from some reference point. Usually (but not always) an angle of 0° from the red axis designates 0 hue, and the hue increases counterclockwise from there. The saturation (distance from the vertical axis) is the length of the vector from the origin to the point. Note that the origin is defined by the intersection of the color plane with the vertical intensity axis. The important components of the HSI color space are the vertical intensity axis, the length of the vector to a color point, and the



a
b c d

FIGURE 6.11
Hue and saturation in the HSI color model. The dot is any color point. The angle from the red axis gives the hue. The length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.

angle this vector makes with the red axis. Therefore, it is not unusual to see the HSI planes defined in terms of the hexagon just discussed, a triangle, or even a circle, as Figs. 6.11(c) and (d) show. The shape chosen does not matter because any one of these shapes can be warped into one of the other two by a geometric transformation. Figure 6.12 shows the HSI model based on color triangles, and on circles.

Converting Colors from RGB to HSI

Computations from RGB to HSI and back are carried out on a pixel-by-pixel basis. We omitted the dependence of the conversion equations on (x,y) for notational clarity.

Given an image in RGB color format, the H component of each RGB pixel is obtained using the equation

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases} \tag{6-16}$$

with[†]

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\} \tag{6-17}$$

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right] \tag{6-18}$$

[†] It is good practice to add a small number in the denominator of this expression to avoid dividing by 0 when $R = G = B$, in which case θ will be 90°. Note that when all RGB components are equal, Eq. (6-18) gives $S = 0$. In addition, the conversion from HSI back to RGB in Eqs. (6-20) through (6-30) will give $R = G = B = I$, as expected, because, when $R = G = B$, we are dealing with a grayscale image.

a
b

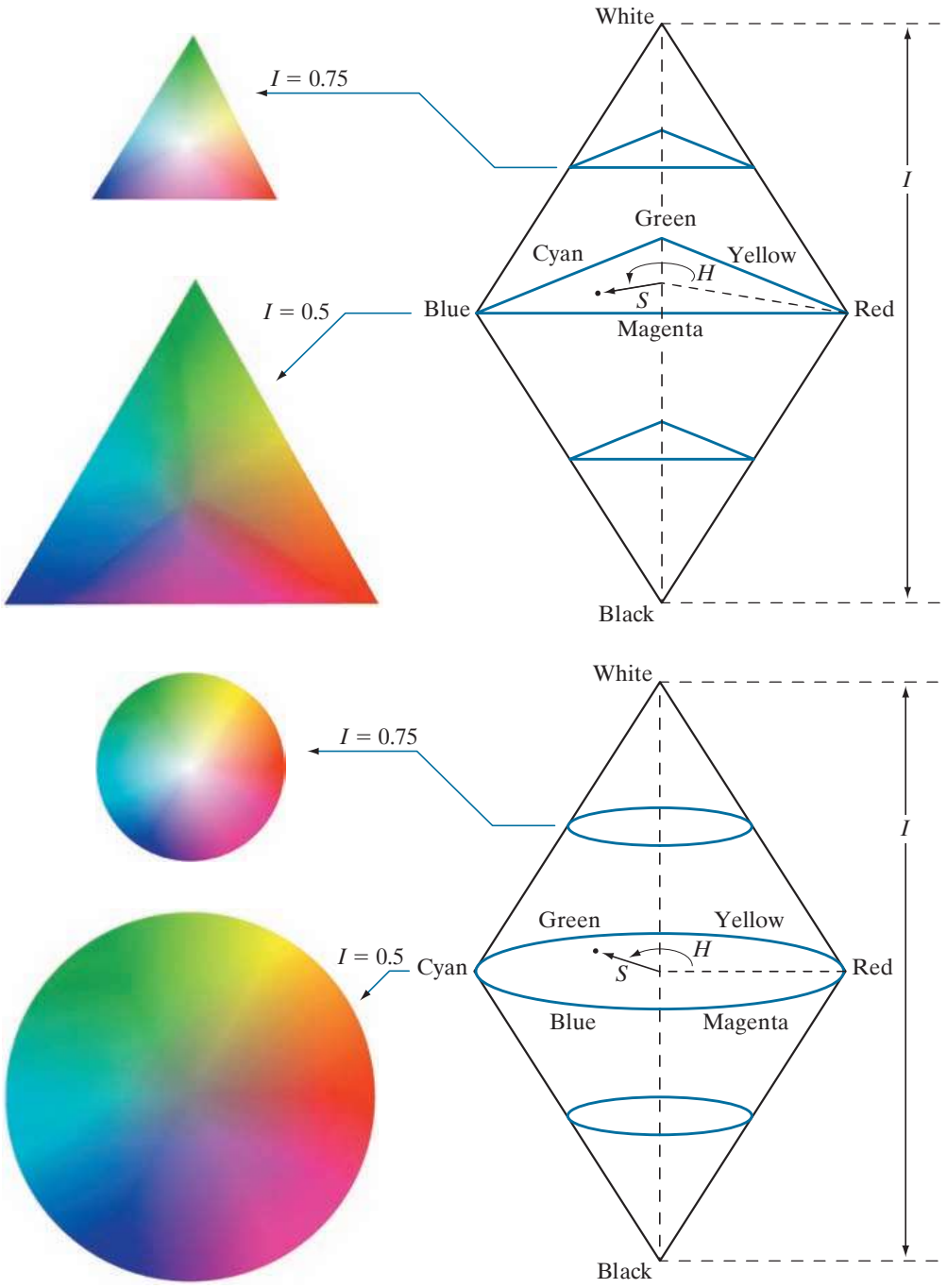


FIGURE 6.12
The HSI color model based on (a) triangular, and (b) circular color planes. The triangles and circles are perpendicular to the vertical intensity axis.

Finally, the intensity component is obtained from the equation

$$I = \frac{1}{3}(R + G + B) \tag{6-19}$$

These equations assume that the RGB values have been normalized to the range [0, 1], and that angle θ is measured with respect to the red axis of the HSI space, as in Fig. 6.11. Hue can be normalized to the range [0, 1] by dividing by 360° all values resulting from Eq. (6-16). The other two HSI components already are in this range if the given RGB values are in the interval [0, 1].

The results in Eqs. (6-16) through (6-19) can be derived from the geometry in Figs. 6.10 and 6.11. The derivation is tedious and would not add significantly to the present discussion. You can find the proof for these equations (and for the equations that follow for HSI to RGB conversion) in the *Tutorials* section of the book website.

Converting Colors from HSI to RGB

Given values of HSI in the interval [0, 1], we now want to find the corresponding RGB values in the same range. The applicable equations depend on the values of H. There are three sectors of interest, corresponding to the 120° intervals in the separation of primaries (see Fig. 6.11). We begin by multiplying H by 360°, which returns the hue to its original range of [0°, 360°].

RG sector $(0° \le H < 120°)$: When H is in this sector, the RGB components are given by the equations

$$B = I(1 - S) \tag{6-20}$$

$$R = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right] \tag{6-21}$$

and

$$G = 3I - (R + B) \tag{6-22}$$

GB sector $(120° \le H < 240°)$: If the given value of H is in this sector, we first subtract 120° from it:

$$H = H - 120° \tag{6-23}$$

Then, the RGB components are

$$R = I(1 - S) \tag{6-24}$$

$$G = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right] \tag{6-25}$$

and

$$B = 3I - (R + G) \tag{6-26}$$

BR sector $(240° \le H \le 360°)$: Finally, if H is in this range, we subtract $240°$ from it:

$$H = H - 240° \tag{6-27}$$

Then, the RGB components are

$$G = I(1 - S) \tag{6-28}$$

$$B = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right] \tag{6-29}$$

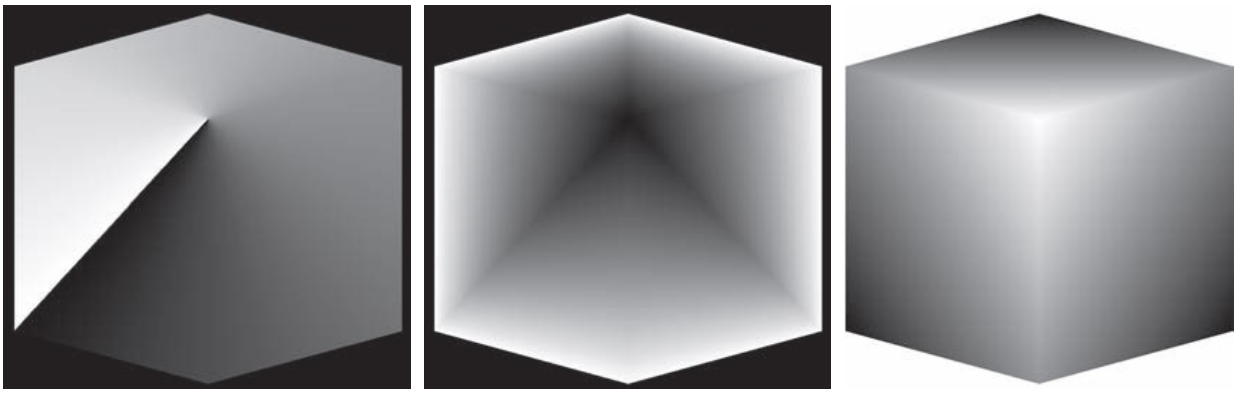
and

$$R = 3I - (G + B) \tag{6-30}$$

We discuss several uses of these equations in the following sections.

EXAMPLE 6.2: The HSI values corresponding to the image of the RGB color cube.

Figure 6.13 shows the hue, saturation, and intensity images for the RGB values in Fig. 6.8. Figure 6.13(a) is the hue image. Its most distinguishing feature is the discontinuity in value along a 45° line in the front (red) plane of the cube. To understand the reason for this discontinuity, refer to Fig. 6.8, draw a line from the red to the white vertices of the cube, and select a point in the middle of this line. Starting at that point, draw a path to the right, following the cube around until you return to the starting point. The major colors encountered in this path are yellow, green, cyan, blue, magenta, and back to red. According to Fig. 6.11, the values of hue along this path should increase from 0° to 360° (i.e., from the lowest to highest



a b c

FIGURE 6.13 HSI components of the image in Fig. 6.8: (a) hue, (b) saturation, and (c) intensity images.

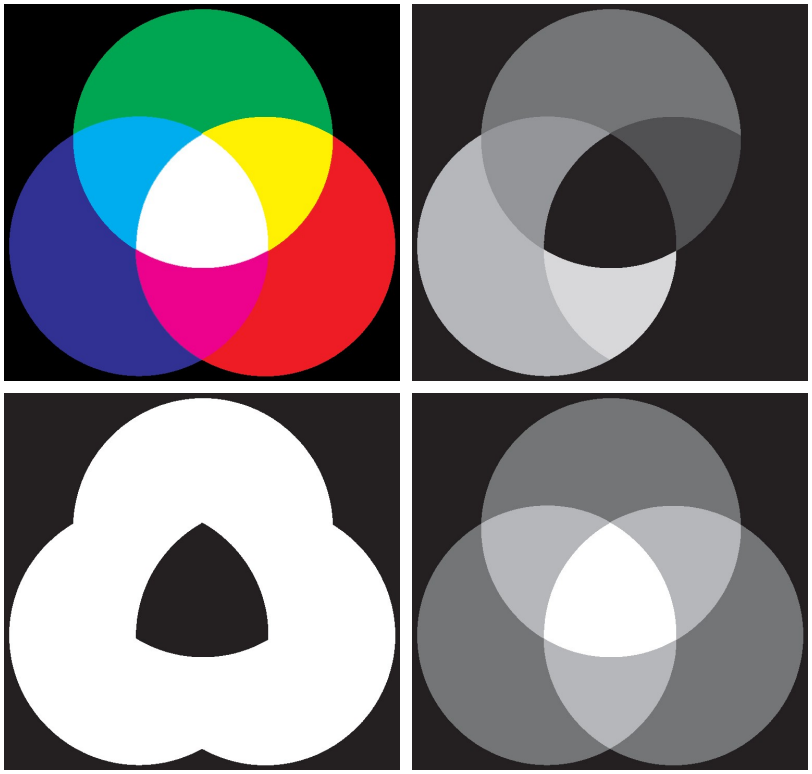
possible values of hue). This is precisely what Fig. 6.13(a) shows, because the lowest value is represented as black and the highest value as white in the grayscale. In fact, the hue image was originally normalized to the range [0, 1] and then scaled to 8 bits; that is, we converted it to the range [0, 255], for display.

The saturation image in Fig. 6.13(b) shows progressively darker values toward the white vertex of the RGB cube, indicating that colors become less and less saturated as they approach white. Finally, every pixel in the intensity image shown in Fig. 6.13(c) is the average of the RGB values at the corresponding pixel in Fig. 6.8.

Manipulating HSI Component Images

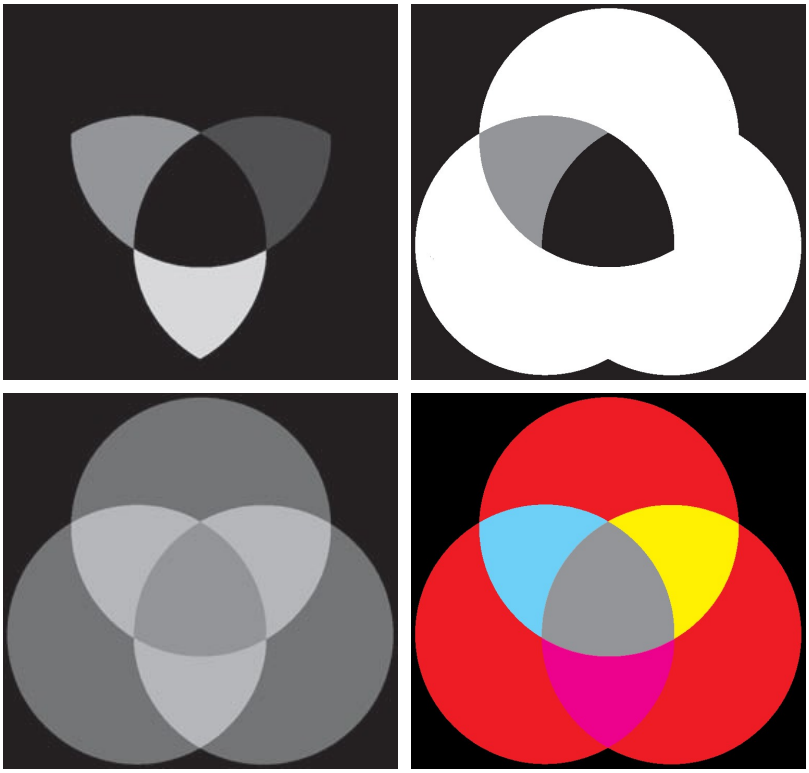
In the following discussion, we take a look at some simple techniques for manipulating HSI component images. This will help you develop familiarity with these comonents, and deepen your understanding of the HSI color model. Figure 6.14(a) shows an image composed of the primary and secondary RGB colors. Figures 6.14(b) through (d) show the H, S, and I components of this image, generated using Eqs. (6-16) through (6-19). Recall from the discussion earlier in this section that the gray-level values in Fig. 6.14(b) correspond to angles; thus, for example, because red corresponds to 0°, the red region in Fig. 6.14(a) is mapped to a black region in the hue image. Similarly, the gray levels in Fig. 6.14(c) correspond to saturation (they were scaled to [0, 255] for display), and the gray levels in Fig. 6.14(d) are average intensities.

To change the individual color of any region in the RGB image, we change the values of the corresponding region in the hue image of Fig. 6.14(b). Then we convert



a b
c d

FIGURE 6.14
(a) RGB image and the components of its corresponding HSI image: (b) hue, (c) saturation, and (d) intensity.



a b
c d

FIGURE 6.15
(a)-(c) Modified HSI component images.
(d) Resulting RGB image. (See Fig. 6.14 for the original HSI images.)

the new H image, along with the unchanged S and I images, back to RGB using the procedure explained in Eqs. (6-20) through (6-30). To change the saturation (purity) of the color in any region, we follow the same procedure, except that we make the changes in the saturation image in HSI space. Similar comments apply to changing the average intensity of any region. Of course, these changes can be made simultaneously. For example, the image in Fig. 6.15(a) was obtained by changing to 0 the pixels corresponding to the blue and green regions in Fig. 6.14(b). In Fig. 6.15(b), we reduced by half the saturation of the cyan region in component image S from Fig. 6.14(c). In Fig. 6.15(c), we reduced by half the intensity of the central white region in the intensity image of Fig. 6.14(d). The result of converting this modified HSI image back to RGB is shown in Fig. 6.15(d). As expected, we see in this figure that the outer portions of all circles are now red; the purity of the cyan region was diminished, and the central region became gray rather than white. Although these results are simple, they clearly illustrate the power of the HSI color model in allowing independent control over hue, saturation, and intensity. These are quantities with which humans are quite familiar when describing colors.

A DEVICE INDEPENDENT COLOR MODEL

As noted earlier, humans see a broad spectrum of colors and color shades. However, color perception differs between individuals. Not only that, but color across devices such as monitors and printers can vary significantly unless these devices are properly calibrated.

Color transformations can be performed on most desktop computers. In conjunction with digital cameras, flatbed scanners, and ink-jet printers, they turn a personal computer into a *digital darkroom*. Also, commercial devices exist that use a combination of spectrometer measurements and software to develop color profiles that can then be loaded on monitors and printers to calibrate their color responses.

The effectiveness of the transformations examined in this section is judged ultimately in print. Because these transformations are developed, refined, and evaluated on monitors, it is necessary to maintain a high degree of color consistency between the monitors used and the eventual output devices. This is best accomplished with a device-independent color model that relates the color gamuts (see Section 6.1) of the monitors and output devices, as well as any other devices being used, to one another. The success of this approach depends on the quality of the color profiles used to map each device to the model, as well as the model itself. The model of choice for many color management systems (CMS) is the CIE $L^*a^*b^*$ model, also called CIELAB (CIE [1978], Robertson [1977]).

The $L^*a^*b^*$ color components are given by the following equations:

$$L^* = 116 \cdot h\left(\frac{Y}{Y_W}\right) - 16 \tag{6-31}$$

$$a^* = 500\left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right)\right] \tag{6-32}$$

and

$$b^* = 200\left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right)\right] \tag{6-33}$$

where

$$h(q) = \begin{cases} \sqrt[3]{q} & q > 0.008856 \\ 7.787q + 16/116 & q \le 0.008856 \end{cases} \tag{6-34}$$

and X_W, Y_W, and Z_W are reference white tristimulus values—typically the white of a perfectly reflecting diffuser under CIE standard D65 illumination (defined by $x = 0.3127$ and $y = 0.3290$ in the CIE chromaticity diagram of Fig. 6.5). The $L^*a^*b^*$ color space is *colorimetric* (i.e., colors perceived as matching are encoded identically), *perceptually uniform* (i.e., color differences among various hues are perceived uniformly—see the classic paper by MacAdams [1942]), and *device independent*. While $L^*a^*b^*$ colors are not directly displayable (conversion to another color space is required), the $L^*a^*b^*$ gamut encompasses the entire visible spectrum and can represent accurately the colors of any display, print, or input device. Like the HSI system, the $L^*a^*b^*$ system is an excellent decoupler of intensity (represented by lightness L^*) and color (represented by a^* for red minus green and b^* for green minus blue), making it useful in both image manipulation (tone and contrast editing) and image compression applications. Studies indicate that the degree to which

the lightness information is separated from the color information in the $L^*a^*b^*$ system is greater than in any other color system (see Kasson and Plouffe [1972]). The principal benefit of calibrated imaging systems is that they allow tonal and color imbalances to be corrected interactively and independently—that is, in two sequential operations. Before color irregularities, like over- and under-saturated colors, are resolved, problems involving the image's tonal range are corrected. The tonal range of an image, also called its *key type*, refers to its general distribution of color intensities. Most of the information in high-key images is concentrated at high (or light) intensities; the colors of low-key images are located predominantly at low intensities; middle-key images lie in between. As in the monochrome case, it is often desirable to distribute the intensities of a color image equally between the highlights and the shadows. In Section 6.4, we give examples showing a variety of color transformations for the correction of tonal and color imbalances.

6.3 PSEUDOCOLOR IMAGE PROCESSING

Pseudocolor (sometimes called *false color*) image processing consists of assigning colors to gray values based on a specified criterion. The term pseudo or false color is used to differentiate the process of assigning colors to achromatic images from the processes associated with true color images, a topic discussed starting in Section 6.4. The principal use of pseudocolor is for human visualization and interpretation of grayscale events in an image or sequence of images. As noted at the beginning of this chapter, one of the principal motivations for using color is the fact that humans can discern thousands of color shades and intensities, compared to less than two dozen shades of gray.

INTENSITY SLICING AND COLOR CODING

The techniques of *intensity* (sometimes called *density*) *slicing* and color coding are the simplest and earliest examples of pseudocolor processing of digital images. If an image is interpreted as a 3-D function [see Fig. 2.18(a)], the method can be viewed as one of placing planes parallel to the coordinate plane of the image; each plane then "slices" the function in the area of intersection. Figure 6.16 shows an example of using a plane at $f(x,y) = l_i$ to slice the image intensity function into two levels.

If a different color is assigned to each side of the plane in Fig. 6.16, any pixel whose intensity level is above the plane will be coded with one color, and any pixel below the plane will be coded with the other. Levels that lie on the plane itself may be arbitrarily assigned one of the two colors, or they could be given a third color to highlight all the pixels at that level. The result is a two- (or three-) color image whose relative appearance can be controlled by moving the slicing plane up and down the intensity axis.

In general, the technique for multiple colors may be summarized as follows. Let $[0, L-1]$ represent the grayscale, let level l_0 represent black $[f(x,y) = 0]$, and level l_{L-1} represent white $[f(x,y) = L-1]$. Suppose that P planes perpendicular to the intensity axis are defined at levels $l_1, l_2, \ldots, l_P$. Then, assuming that $0 < P < L-1$, the P planes partition the grayscale into $P+1$ intervals, $I_1, I_2, \ldots, I_{P+1}$. Intensity to color assignments at each pixel location (x, y) are made according to the equation

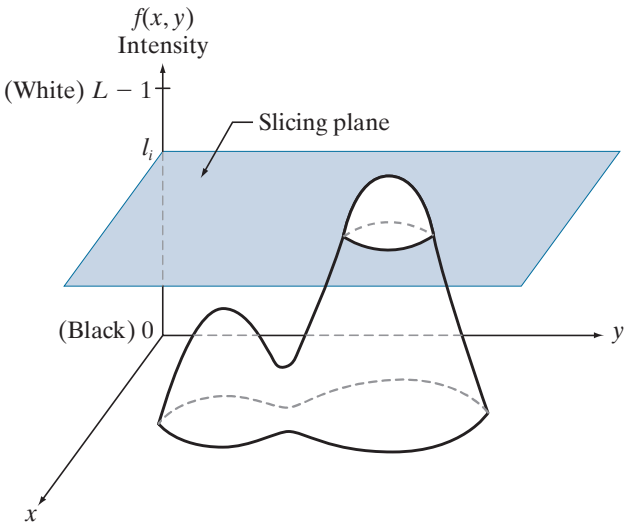


FIGURE 6.16 Graphical interpretation of the intensity-slicing technique.

$$\text{if } f(x,y) \in I_k, \text{ let } f(x,y) = c_k \tag{6-35}$$

where c_k is the color associated with the kth intensity interval I_k, defined by the planes at $l = k-1$ and $l = k$.

Figure 6.16 is not the only way to visualize the method just described. Figure 6.17 shows an equivalent approach. According to the mapping in this figure, any image intensity below level l_i is assigned one color, and any level above is assigned another. When more partitioning levels are used, the mapping function takes on a staircase form.

EXAMPLE 6.3: Intensity slicing and color coding.

A simple but practical use of intensity slicing is shown in Fig. 6.18. Figure 6.18(a) is a grayscale image of the Picker Thyroid Phantom (a radiation test pattern), and Fig. 6.18(b) is the result of intensity slicing this image into eight colors. Regions that appear of constant intensity in the grayscale image are actually quite variable, as shown by the various colors in the sliced image. For instance, the left lobe is a dull gray in the grayscale image, and picking out variations in intensity is difficult. By contrast, the color image

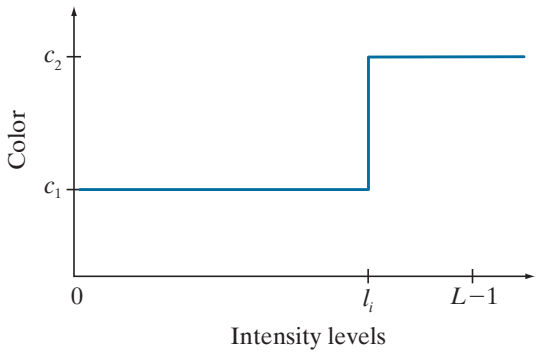
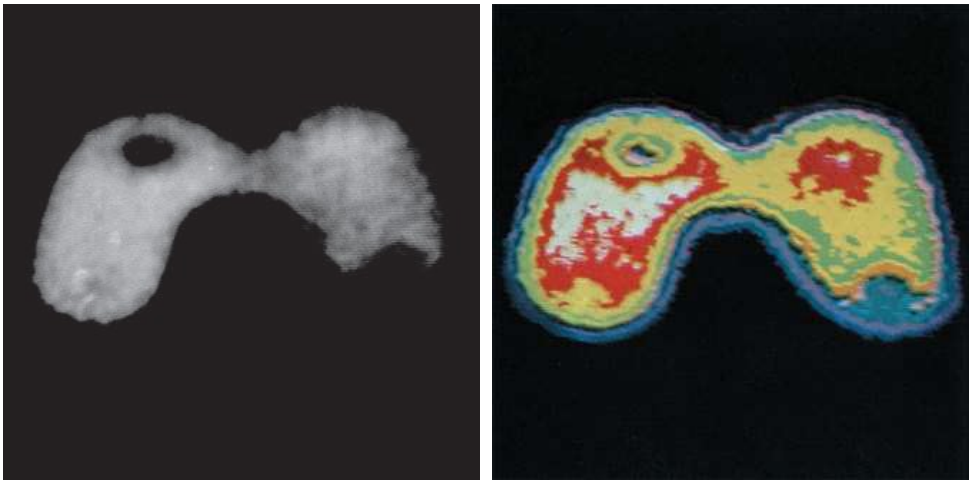


FIGURE 6.17 An alternative representation of the intensity-slicing technique.

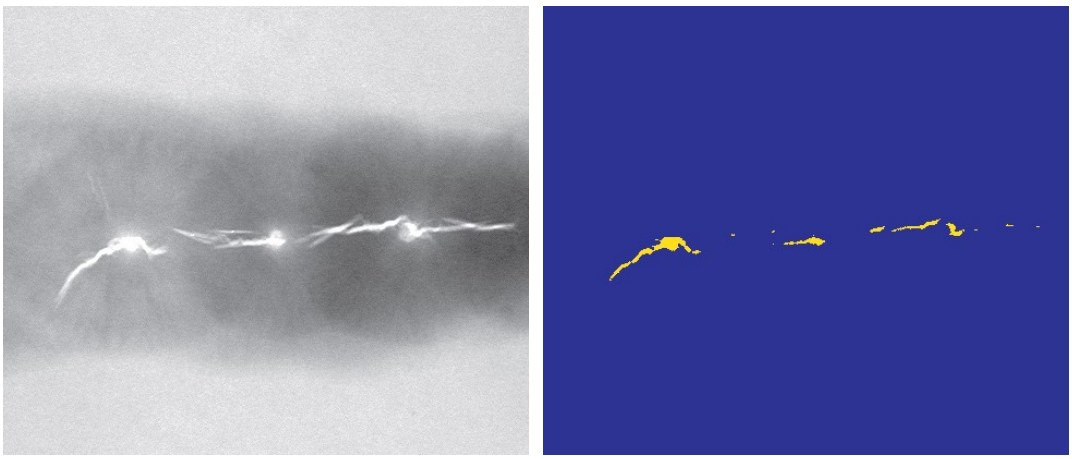


a b

FIGURE 6.18
(a) Grayscale image of the Picker Thyroid Phantom. (b) Result of intensity slicing using eight colors. (Courtesy of Dr. J. L. Blankenship, Oak Ridge National Laboratory.)

clearly shows eight different regions of constant intensity, one for each of the colors used. By varying the number of colors and the span of the intensity intervals, one can quickly determine the characteristics of intensity variations in a grayscale image. This is particularly true in situations such as the one shown here, in which the object of interest has uniform texture with intensity variations that are difficult to analyze visually. This example also illustrates the comments made in Section 6.1 about the eye's superior capability for detecting different color shades.

In the preceding simple example, the grayscale was divided into intervals and a different color was assigned to each, with no regard for the meaning of the gray levels in the image. Interest in that case was simply to view the different gray levels constituting the image. Intensity slicing assumes a much more meaningful and useful role when subdivision of the grayscale is based on physical characteristics of the image. For instance, Fig. 6.19(a) shows an X-ray image of a weld (the broad, horizontal dark region) containing several cracks and porosities (the bright streaks running horizontally through the middle of the image). When there is a porosity or crack in a weld, the full strength of the X-rays going through the object saturates the imaging sensor on the other side of the object. Thus, intensity values of 255 in an 8-bit image coming from such a system automatically imply a problem with the weld. If human visual analysis is used to inspect welds (still a common procedure today), a simple color coding that assigns



a b

FIGURE 6.19
(a) X-ray image of a weld. (b) Result of color coding. (Original image courtesy of X-TEK Systems, Ltd.)

one color to level 255 and another to all other intensity levels can simplify the inspector's job considerably. Figure 6.19(b) shows the result. No explanation is required to arrive at the conclusion that human error rates would be lower if images were displayed in the form of Fig. 6.19(b), instead of the form in Fig. 6.19(a). In other words, if an intensity value, or range of values, one is looking for is known, intensity slicing is a simple but powerful aid in visualization, especially if numerous images have to be inspected on a routine basis.

EXAMPLE 6.4: Use of color to highlight rainfall levels.

Measurement of rainfall levels, especially in the tropical regions of the Earth, is of interest in diverse applications dealing with the environment. Accurate measurements using ground-based sensors are difficult and expensive to acquire, and total rainfall figures are even more difficult to obtain because a significant portion of precipitation occurs over the ocean. One approach for obtaining rainfall figures remotely is to use satellites. The TRMM (Tropical Rainfall Measuring Mission) satellite utilizes, among others, three sensors specially designed to detect rain: a precipitation radar, a microwave imager, and a visible and infrared scanner (see Sections 1.3 and 2.3 regarding image sensing modalities).

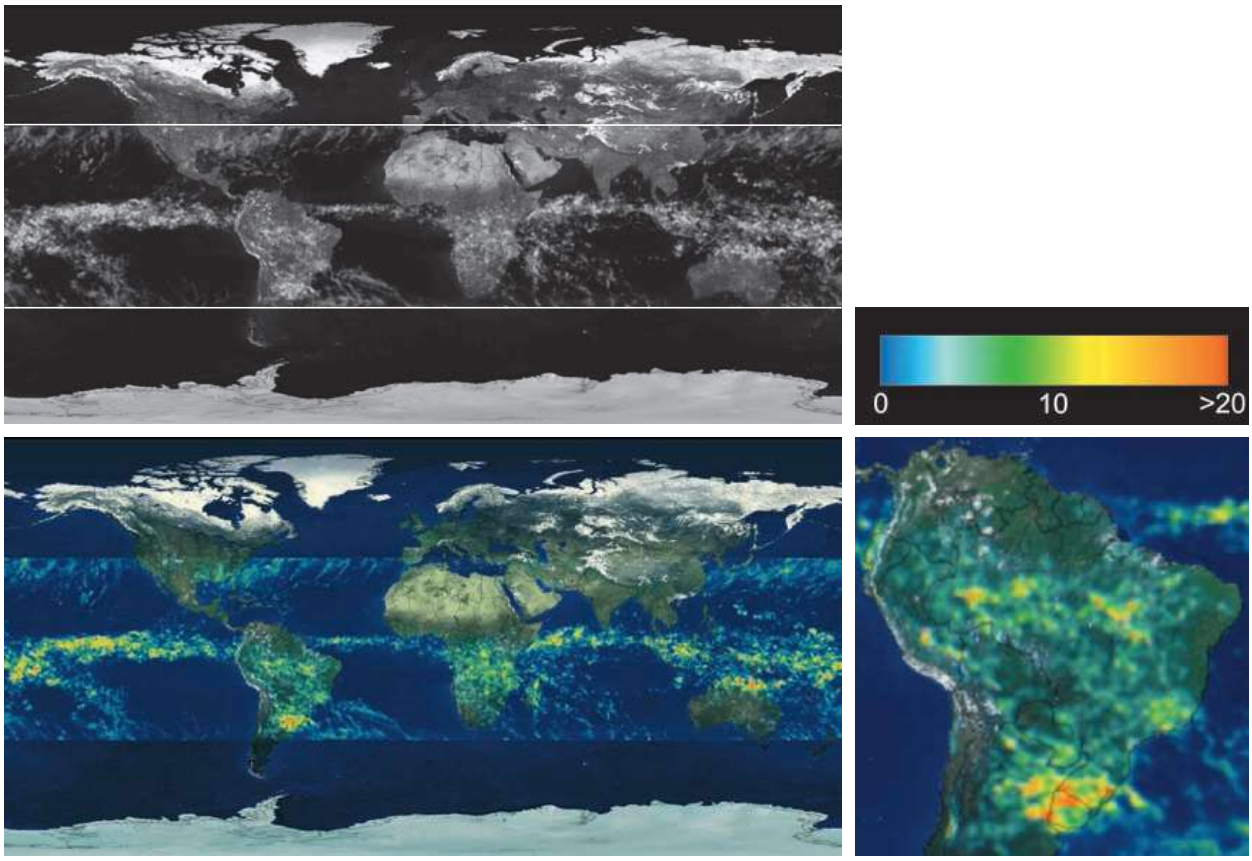
The results from the various rain sensors are processed, resulting in estimates of average rainfall over a given time period in the area monitored by the sensors. From these estimates, it is not difficult to generate grayscale images whose intensity values correspond directly to rainfall, with each pixel representing a physical land area whose size depends on the resolution of the sensors. Such an intensity image is shown in Fig. 6.20(a), where the area monitored by the satellite is the horizontal band highlighted in the middle of the picture (these are tropical regions). In this particular example, the rainfall values are monthly averages (in inches) over a three-year period.

Visual examination of this picture for rainfall patterns is difficult and prone to error. However, suppose that we code intensity levels from 0 to 255 using the colors shown in Fig. 6.20(b). In this mode of intensity slicing, each slice is one of the colors in the color band. Values toward the blues signify low values of rainfall, with the opposite being true for red. Note that the scale tops out at pure red for values of rainfall greater than 20 inches. Figure 6.20(c) shows the result of color coding the grayscale image with the color map just discussed. The results are much easier to interpret, as shown in this figure and in the zoomed area of Fig. 6.20(d). In addition to providing global coverage, this type of data allows meteorologists to calibrate ground-based rain monitoring systems with greater precision than ever before.

INTENSITY TO COLOR TRANSFORMATIONS

Other types of transformations are more general, and thus are capable of achieving a wider range of pseudocolor enhancement results than the simple slicing technique discussed in the preceding section. Figure 6.21 shows an approach that is particularly attractive. Basically, the idea underlying this approach is to perform three independent transformations on the intensity of input pixels. The three results are then fed separately into the red, green, and blue channels of a color monitor. This method produces a composite image whose color content is modulated by the nature of the transformation functions.

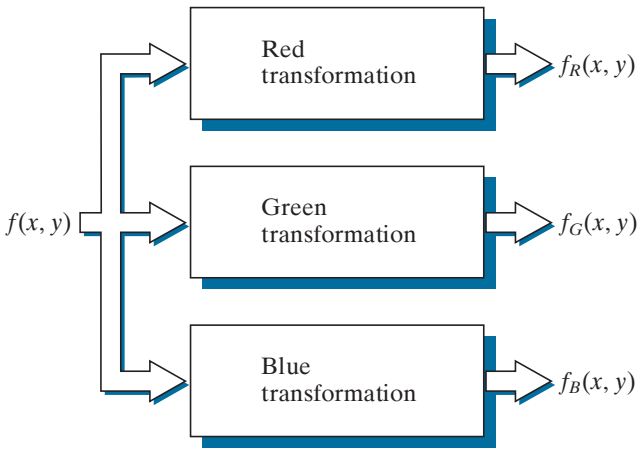
The method for intensity slicing discussed in the previous section is a special case of the technique just described. There, piecewise linear functions of the intensity levels (see Fig. 6.17) are used to generate colors. On the other hand, the method

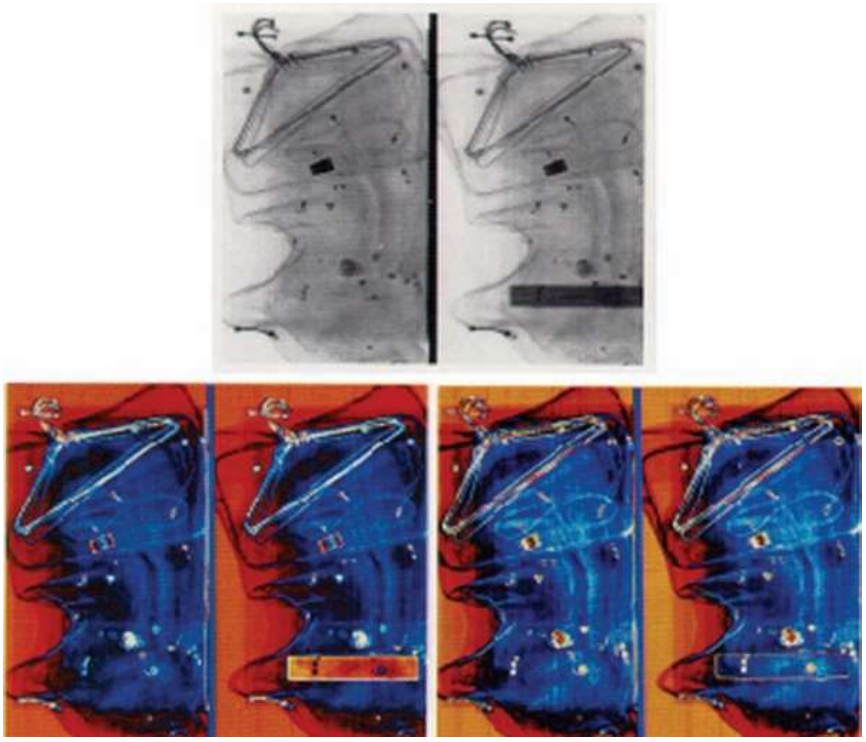


a b
c d

FIGURE 6.20 (a) Grayscale image in which intensity (in the horizontal band shown) corresponds to average monthly rainfall. (b) Colors assigned to intensity values. (c) Color-coded image. (d) Zoom of the South American region. (Courtesy of NASA.)

FIGURE 6.21 Functional block diagram for pseudocolor image processing. Images f_R, f_G,and f_B are fed into the corresponding red, green, and blue inputs of an RGB color monitor.





a
b c

FIGURE 6.22
Pseudocolor enhancement by using the gray level to color transformations in Fig. 6.23. (Original image courtesy of Dr. Mike Hurwitz, Westinghouse.)

discussed in this section can be based on smooth, nonlinear functions, which gives the technique considerable flexibility.

EXAMPLE 6.5: Using pseudocolor to highlight explosives in X-ray images.

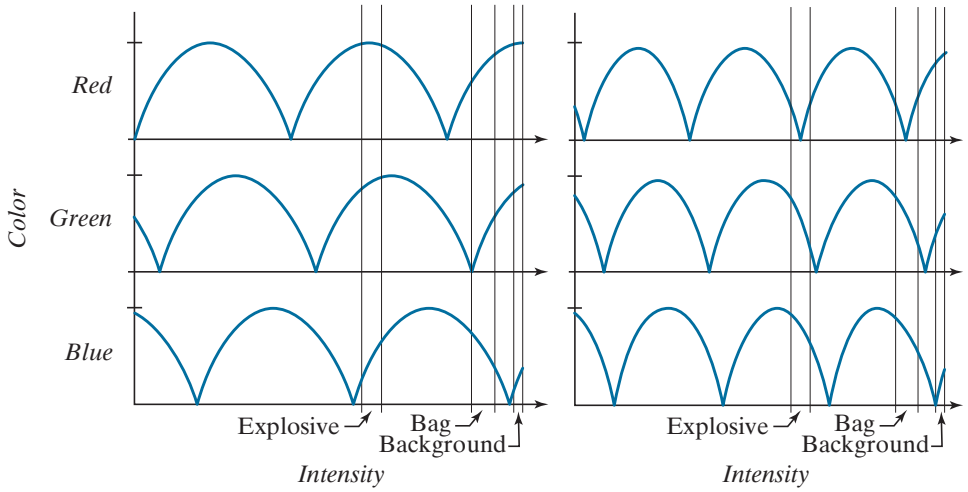
Figure 6.22(a) shows two monochrome images of luggage obtained from an airport X-ray scanning system. The image on the left contains ordinary articles. The image on the right contains the same articles, as well as a block of simulated plastic explosives. The purpose of this example is to illustrate the use of intensity to color transformations to facilitate detection of the explosives.

Figure 6.23 shows the transformation functions used. These sinusoidal functions contain regions of relatively constant value around the peaks as well as regions that change rapidly near the valleys. Changing the phase and frequency of each sinusoid can emphasize (in color) ranges in the grayscale. For instance, if all three transformations have the same phase and frequency, the output will be a grayscale image. A small change in the phase between the three transformations produces little change in pixels whose intensities correspond to peaks in the sinusoids, especially if the sinusoids have broad profiles (low frequencies). Pixels with intensity values in the steep section of the sinusoids are assigned a much stronger color content as a result of significant differences between the amplitudes of the three sinusoids caused by the phase displacement between them.

The image in Fig. 6.22(b) was obtained using the transformation functions in Fig. 6.23(a), which shows the gray-level bands corresponding to the explosive, garment bag, and background, respectively. Note that the explosive and background have quite different intensity levels, but they were both coded with approximately the same color as a result of the periodicity of the sine waves. The image in Fig. 6.22(c) was obtained with the transformation functions in Fig. 6.23(b). In this case, the explosives and garment bag intensity bands were mapped by similar transformations, and thus received essentially the same

a b

FIGURE 6.23 Transformation functions used to obtain the pseudocolor images in Fig. 6.22.



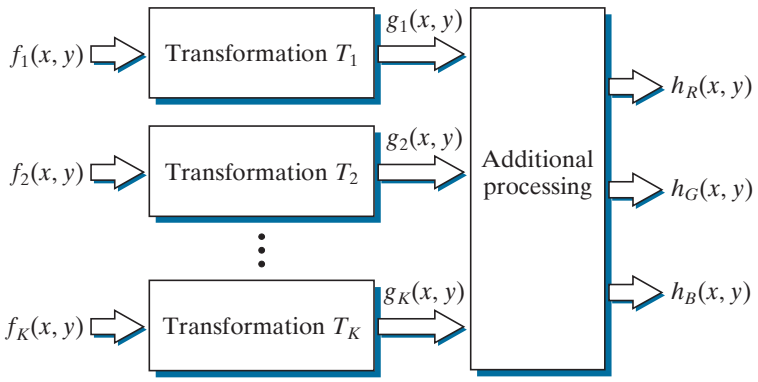
color assignments. Note that this mapping allows an observer to "see" through the explosives. The background mappings were about the same as those used for Fig. 6.22(b), producing almost identical color assignments for the two pseudocolor images.

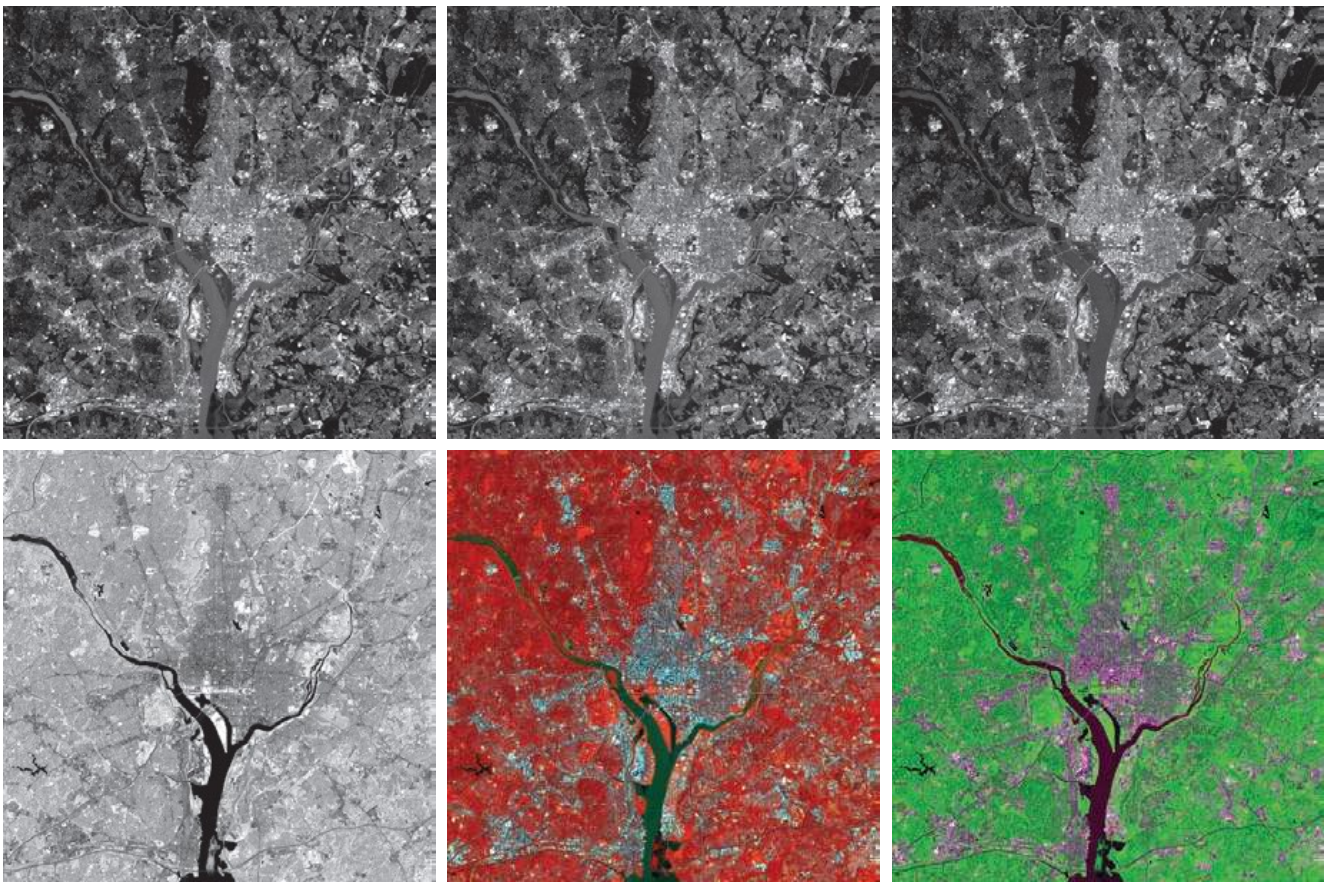
The approach in Fig. 6.21 is based on a single grayscale image. Often, it is of interest to combine several grayscale images into a single color composite, as illustrated in Fig. 6.24. A frequent use of this approach is in multispectral image processing, where different sensors produce individual grayscale images, each in a different spectral band (see Example 6.6 below). The types of additional processing shown in Fig. 6.24 can be techniques such as color balancing and spatial filtering, as discussed later in this chapter. When coupled with background knowledge about the physical characteristics of each band, color-coding in the manner just explained is a powerful aid for human visual analysis of complex multispectral images.

EXAMPLE 6.6: Color coding of multispectral images.

Figures 6.25(a) through (d) show four satellite images of the Washington, D.C., area, including part of the Potomac River. The first three images are in the visible red (R), green (G), and blue (B) bands, and

FIGURE 6.24 A pseudocolor coding approach using multiple grayscale images. The inputs are grayscale images. The outputs are the three components of an RGB composite image.





a b c
d e f

FIGURE 6.25 (a)–(d) Red (R), green (G), blue (B), and near-infrared (IR) components of a LANDSAT multispectral image of the Washington, D.C. area. (e) RGB color composite image obtained using the IR, G, and B component images. (f) RGB color composite image obtained using the R, IR, and B component images. (Original multispectral images courtesy of NASA.)

the fourth is in the near infrared (IR) band (see Table 1.1 and Fig. 1.10). The latter band is responsive to the biomass content of a scene, and we want to use this fact to create a composite RGB color image in which vegetation is emphasized and the other components of the scene are displayed in more muted tones.

Figure 6.25(e) is an RGB composite obtained by replacing the red image by infrared. As you see, vegetation shows as a bright red, and the other components of the scene, which had a weaker response in the near-infrared band, show in pale shades of blue-green. Figure 6.25(f) is a similar image, but with the green replaced by infrared. Here, vegetation shows in a bright green color, and the other components of the scene show in purplish color shades, indicating that their major components are in the red and blue bands. Although the last two images do not introduce any new physical information, these images are much easier to interpret visually once it is known that the dominant component of the images are pixels of areas heavily populated by vegetation.

The type of processing just illustrated uses the physical characteristics of a single band in a multispectral image to emphasize areas of interest. The same approach can help visualize events of interest

a
b

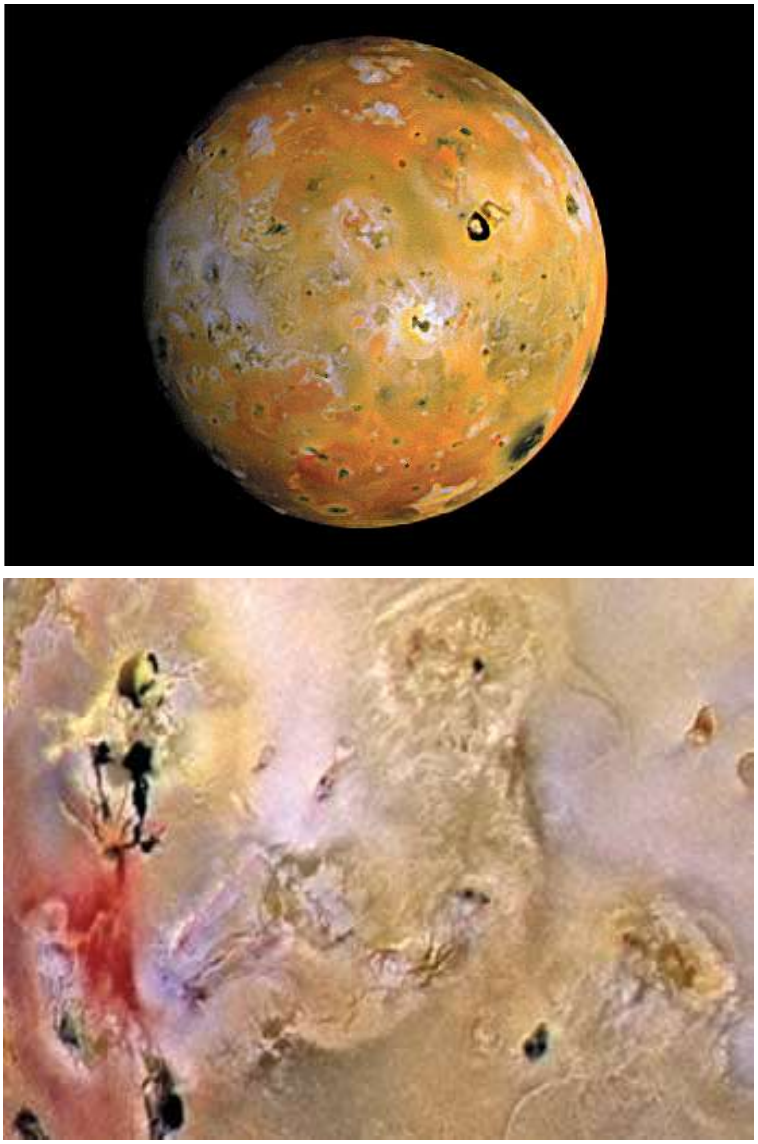


FIGURE 6.26
(a) Pseudocolor rendition of Jupiter Moon Io. (b) A close-up. (Courtesy of NASA.)

in complex images in which the events are beyond human visual sensing capabilities. Figure 6.26 is an excellent illustration of this. These are images of the Jupiter moon Io, shown in pseudocolor by combining several of the sensor images from the Galileo spacecraft, some of which are in spectral regions not visible to the eye. However, by understanding the physical and chemical processes likely to affect sensor response, it is possible to combine the sensed images into a meaningful pseudocolor map. One way to combine the sensed image data is by how they show either differences in surface chemical composition or changes in the way the surface reflects sunlight. For example, in the pseudocolor image in Fig. 6.26(b), bright red depicts material newly ejected from an active volcano on Io, and the surrounding yellow materials are older sulfur deposits. This image conveys these characteristics much more readily than would be possible by analyzing the component images individually.

6.4 BASICS OF FULL-COLOR IMAGE PROCESSING

In this section, we begin the study of processing methods for full-color images. The techniques developed in the sections that follow are illustrative of how full-color images are handled for a variety of image processing tasks. Full-color image processing approaches fall into two major categories. In the first category, we process each grayscale component image individually, then form a composite color image from the individually processed components. In the second category, we work with color pixels directly. Because full-color images have at least three components, color pixels are vectors. For example, in the RGB system, each color point can be interpreted as a vector extending from the origin to that point in the RGB coordinate system (see Fig. 6.7).

Let $\mathbf{c}$ represent an arbitrary vector in RGB color space:

$$\mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{6-36}$$

This equation indicates that the components of $\mathbf{c}$ are the RGB components of a color image at a point. We take into account the fact that the colors of the pixels in an image are a function of spatial coordinates (x, y) by using the notation

Although an RGB image is composed of three grayscale component images, pixels in all three images are registered spatially. That is, a *single* pair of spatial coordinates, (x, y), addresses the *same* pixel location in all three images, as illustrated in Fig. 6.27(b) below.

$$\mathbf{c}(x,y) = \begin{bmatrix} c_R(x,y) \\ c_G(x,y) \\ c_B(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix} \tag{6-37}$$

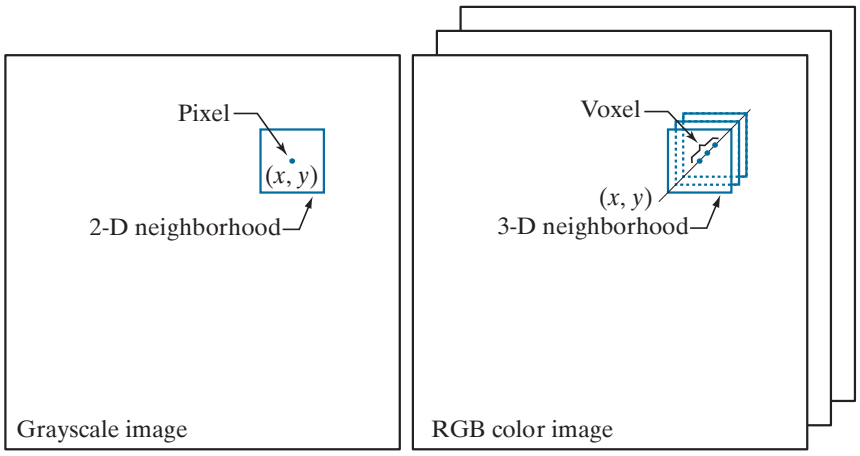
For an image of size $M \times N$, there are MN such vectors, $\mathbf{c}(x, y)$, for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$.

Equation (6-37) depicts a vector whose components are *spatial* variables x and y. This is a frequent source of confusion that can be avoided by focusing on the fact that our interest lies in spatial processes. That is, we are interested in image processing techniques formulated in x and y. The fact that the pixels are now color pixels introduces a factor that, in its easiest formulation, allows us to process a color image by processing each of its component images separately, using standard grayscale image processing methods. However, the results of individual color component processing are not always equivalent to direct processing in color vector space, in which case we must use approaches for processing the elements of color points directly. When these points have more than two components, we call them *voxels*. We use the terms vectors, points, and voxels interchangeably when the meaning is clear that we are referring to images composed of more than one 2-D image.

In order for per-component-image and vector-based processing to be equivalent, two conditions have to be satisfied: first, the process has to be applicable to both vectors and scalars; second, the operation on each component of a vector (i.e., each voxel) must be independent of the other components. As an illustration, Fig. 6.27 shows spatial neighborhood processing of grayscale and full-color images. Suppose

a b

FIGURE 6.27 Spatial neighborhoods for grayscale and RGB color images. Observe in (b) that a *single* pair of spatial coordinates, (x, y), addresses the same spatial location in all three images.



that the process is neighborhood averaging. In Fig. 6.27(a), averaging would be done by summing the intensities of all the pixels in the 2-D neighborhood, then dividing the result by the total number of pixels in the neighborhood. In Fig. 6.27(b), averaging would be done by summing all the voxels in the 3-D neighborhood, then dividing the result by the total number of voxels in the neighborhood. Each of the three component of the average voxel is the sum of the pixels in the single image neighborhood centered on that location. But the same result would be obtained if the averaging were done on the pixels of each image, independently, and then the sum of the three values were added for each. Thus, spatial neighborhood averaging can be carried out on a per-component-image or directly on RGB image voxels. The results would be the same. In the following sections we develop methods for which the per-component-image approach is suitable, and methods for which it is not.

6.5 COLOR TRANSFORMATIONS

The techniques described in this section, collectively called *color transformations*, deal with processing the components of a color image within the context of a *single* color model, as opposed to color transformations *between* color models, as in Section 6.2.

FORMULATION

As with the intensity transformation techniques of Chapter 3, we model color transformations for multispectral images using the general expression

$$s_i = T_i(r_i) \qquad i = 1, 2, \ldots, n \tag{6-38}$$

where n is the total number of component images, r_i are the intensity values of the input component images, s_i are the spatially corresponding intensities in the output component images, and T_i are a set of *transformation* or *color mapping functions* that operate on r_i to produce s_i. Equation (6-38) is applied individually to all pixels in the input image. For example, in the case of RGB color images, $n = 3$, r_1, r_2, r_3 are the intensities values at a point in the input components images, and s_1, s_2, s_3 are

the corresponding transformed pixels in the output image. The fact that i is also a subscript on T means that, in principle, we can implement a different transformation for each input component image.

As an illustration, the first row of Fig. 6.28 shows a full color CMYK image of a simple scene, and the second row shows its four component images, all normalized to the range [0, 1]. We see that the strawberries are composed of large amounts of magenta and yellow because the images corresponding to these two CMYK components are the brightest. Black is used sparingly and is generally confined to the coffee and shadows within the bowl of strawberries. The fourth row shows the equivalent RGB images obtained from the CMYK images using Eqs. (6-13)-(6-15). Here we see that the strawberries contain a large amount of red and very little (although some) green and blue. From the RGB images, we obtained the CMY images in the third row using Eq. (6-5). Note that these CMY images are slightly different from the CMY images in the row above them. This is because the CMY images in these two systems are different as a result of using K in one of them. The last row of Fig. 6.28 shows the HSI components, obtained from the RGB images using Eqs. (6-16)-(6-19). As expected, the intensity (I) component is a grayscale rendition of the full-color original. The saturation image (S) is as expected also. The strawberries are relatively pure in color; as a result, they show the highest saturation (least dilution by white light) values of any of the other elements of the image. Finally, we note some difficulty in interpreting the values of the hue (H) component image. The problem is that (1) there is a discontinuity in the HSI model where 0° and 360° meet [see Fig. 6.13(a)], and (2) hue is undefined for a saturation of 0 (i.e., for white, black, and pure grays). The discontinuity of the model is most apparent around the strawberries, which are depicted in gray level values near both black (0) and white (1). The result is an unexpected mixture of highly contrasting gray levels to represent a single color—red.

We can apply Eq. (6-38) to any of the color-space component images in Fig. 6.28. In theory, any transformation can be performed in any color model. In practice, however, some operations are better suited to specific models. For a given transformation, the effects of converting between representations must be factored into the decision regarding the color space in which to implement it. For example, suppose that we wish to modify the intensity of the full-color image in the first row of Fig. 6.28 by a constant value, k in the range [0, 1]. In the HSI color space we need to modify only the intensity component image:

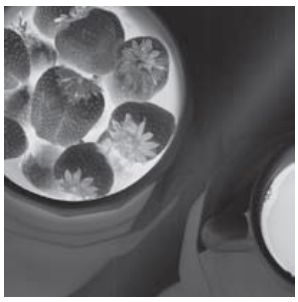
$$s_3 = kr_3 \tag{6-39}$$

and we let $s_1 = r_1$ and $s_2 = r_2$. In terms of our earlier discussion note that we are using two different transformation functions: T_1 and T_2 are identity transformations, and T_3 is a constant transformation.

In the RGB color space we need to modify all three components by the same constant transformation:

$$s_i = kr_i \qquad i = 1,2,3 \tag{6-40}$$

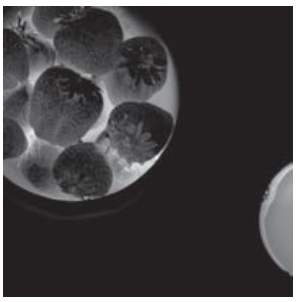
Full color image



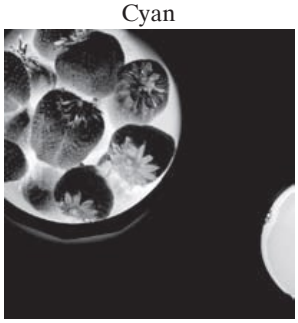
Cyan

Magenta

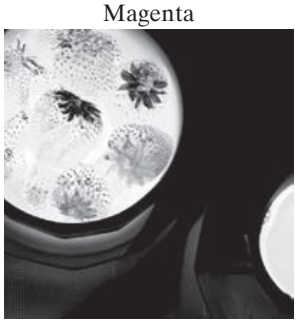
Yellow



Black

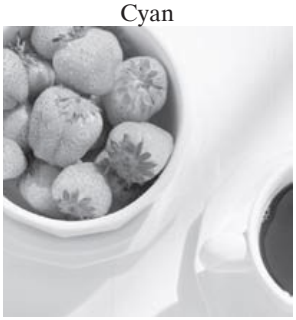


Cyan



Magenta

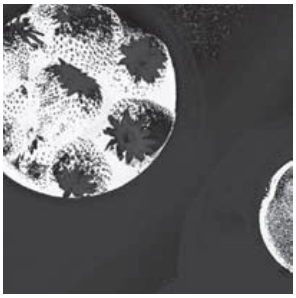
Yellow



Red

Green

Blue



Hue

Saturation

Intensity

FIGURE 6.28 A full-color image and its various color-space components. (Original image courtesy of MedData Interactive.)

The CMY space requires a similar set of linear transformations (see Problem 6.16):

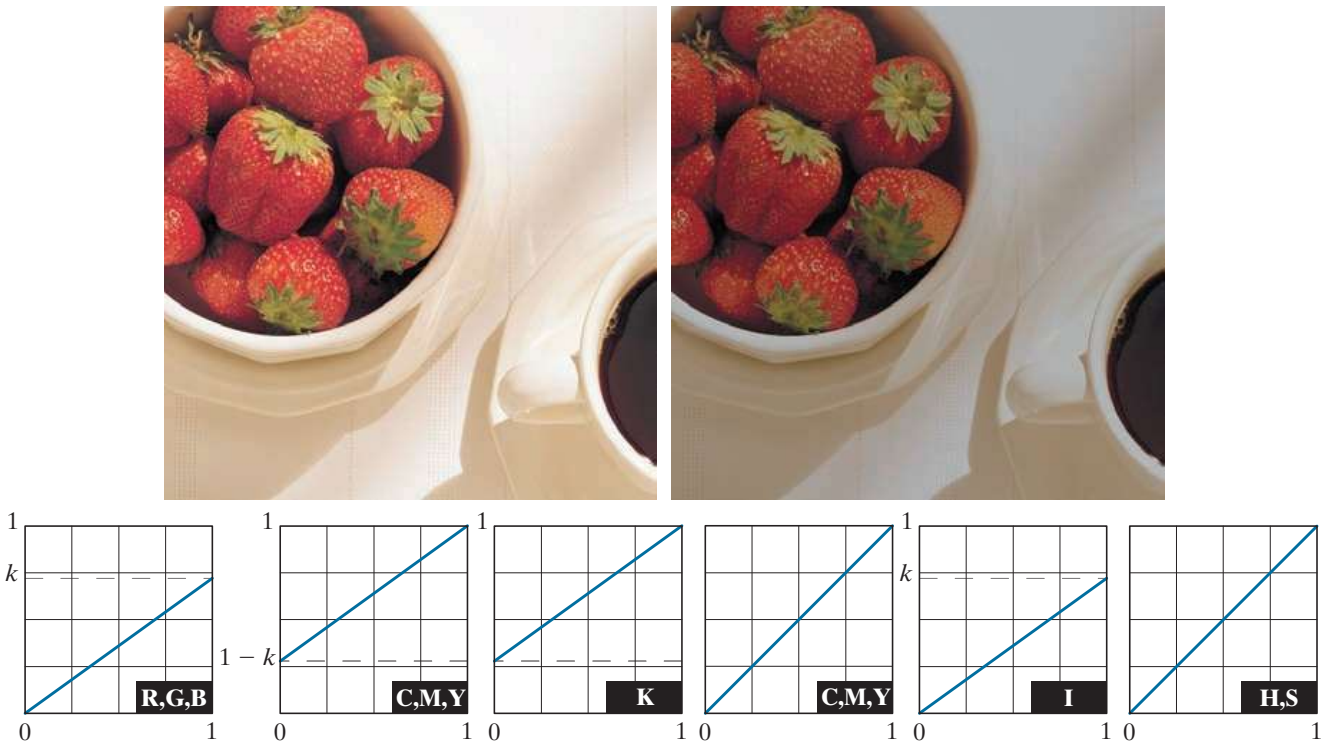
$$s_i = kr_i + (1-k) \qquad i = 1,2,3 \tag{6-41}$$

Similarly, the transformations required to change the intensity of the CMYK image is given by

$$s_i = \begin{cases} r_i & i = 1,2,3 \\ kr_i + (1-k) & i = 4 \end{cases} \tag{6-42}$$

This equation tells us that to change the intensity of a CMYK image, we only change the fourth (K) component.

Figure 6.29(b) shows the result of applying the transformations in Eqs. (6-39) through (6-42) to the full-color image of Fig. 6.28, using $k = 0.7$. The mapping functions themselves are shown graphically in Figs. 6.29(c) through (h). Note that the mapping function for CMYK consist of two parts, as do the functions for HSI; one of the transformations handles one component, and the other does the rest. Although



a b
c d e f g h

FIGURE 6.29 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c) The required RGB mapping function. (d)–(e) The required CMYK mapping functions. (f) The required CMY mapping function. (g)–(h) The required HSI mapping functions. (Original image courtesy of MedData Interactive.)

we used several different transformations, the net result of changing the intensity of the color by a constant value was the same for all.

It is important to note that each transformation defined in Eqs. (6-39) through (6-42) depends only on one component within its color space. For example, the red output component, s_1, in Eq. (6-40) is independent of the green (r_2) and blue (r_3) inputs; it depends only on the red (r_1) input. Transformations of this type are among the simplest and most frequently used color processing tools. They can be carried out on a per-color-component basis, as mentioned at the beginning of our discussion. In the remainder of this section, we will examine several such transformations and discuss a case in which the component transformation functions are dependent on all the color components of the input image and, therefore, cannot be done on an individual color-component basis.

COLOR COMPLEMENTS

The *color circle* (also called the *color wheel*) shown in Fig. 6.30 originated with Sir Isaac Newton, who in the seventeenth century created its first form by joining the ends of the color spectrum. The color circle is a visual representation of colors that are arranged according to the chromatic relationship between them. The circle is formed by placing the primary colors equidistant from each other. Then, the secondary colors are placed between the primaries, also in an equidistant arrangement. The net result is that hues directly opposite one another on the color circle are *complements*. Our interest in complements stems from the fact that they are analogous to the grayscale negatives we studied in Section 3.2. As in the grayscale case, color complements are useful for enhancing detail that is embedded in dark regions of a color image—particularly when the regions are dominant in size. The following example illustrates some of these concepts.

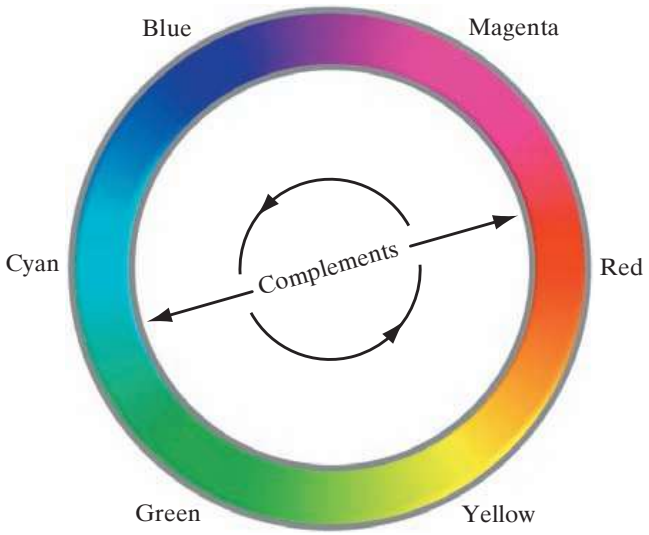
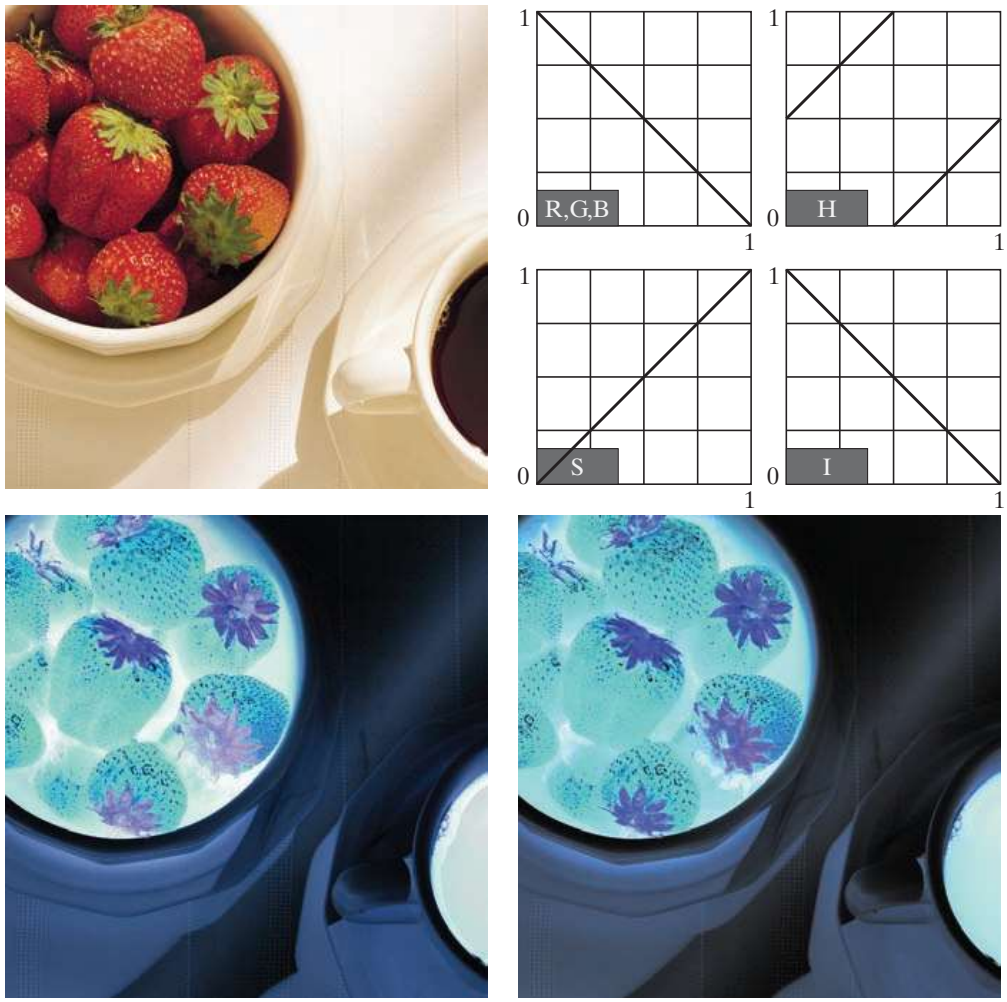


FIGURE 6.30 Color complements on the color circle.

EXAMPLE 6.7: Computing color image complements.

Figures 6.31(a) and (c) show the full-color image from Fig. 6.28 and its color complement. The RGB transformations used to compute the complement are plotted in Fig. 6.31(b). They are identical to the grayscale negative transformation defined in Section 3.2. Note that the complement is reminiscent of conventional photographic color film negatives. Reds of the original image are replaced by cyans in the complement. When the original image is black, the complement is white, and so on. Each of the hues in the complement image can be predicted from the original image using the color circle of Fig. 6.30, and each of the RGB component transforms involved in the computation of the complement is a function of only the corresponding input color component.

Unlike the intensity transformations of Fig. 6.29, the RGB complement transformation functions used in this example do not have a straightforward HSI equivalent. It is left as an exercise (see Problem 6.19) to show that the saturation component of the complement cannot be computed from the saturation component of the input image alone. Figure 6.31(d) shows an approximation of the complement using the hue, saturation, and intensity transformations in Fig. 6.31(b). The saturation component of the input image is unaltered; it is responsible for the visual differences between Figs. 6.31(c) and (d).



a b
c d

FIGURE 6.31 Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

COLOR SLICING

Highlighting a specific range of colors in an image is useful for separating objects from their surroundings. The basic idea is either to: (1) display the colors of interest so that they stand out from the background; or (2) use the region defined by the colors as a mask for further processing. The most straightforward approach is to extend the intensity slicing techniques of Section 3.2. However, because a color pixel is an n-dimensional quantity, the resulting color transformation functions are more complicated than their grayscale counterparts in Fig. 3.11. In fact, the required transformations are more complex than the color component transforms considered thus far. This is because all practical color-slicing approaches require each pixel's transformed color components to be a function of all n original pixel's color components.

One of the simplest ways to "slice" a color image is to map the colors outside some range of interest into a nonprominent neutral color. If the colors of interest are enclosed by a cube (or *hypercube* for $n > 3$) of width W and centered at a prototypical (e.g., average) color with components $(a_1, a_2, \ldots, a_n)$, the necessary set of transformations are given by

$$s_i = \begin{cases} 0.5 & \text{if } \left[|r_j - a_j| > \frac{W}{2} \right]_{\text{any } 1 \le j \le n} \\ r_i & \text{otherwise} \end{cases} \qquad i = 1, 2, \ldots, n \tag{6-43}$$

These transformations highlight the colors around the prototype by forcing all other colors to the midpoint of the reference color space (this is an arbitrarily chosen neutral point). For the RGB color space, for example, a suitable neutral point is middle gray or color (0.5, 0.5, 0.5).

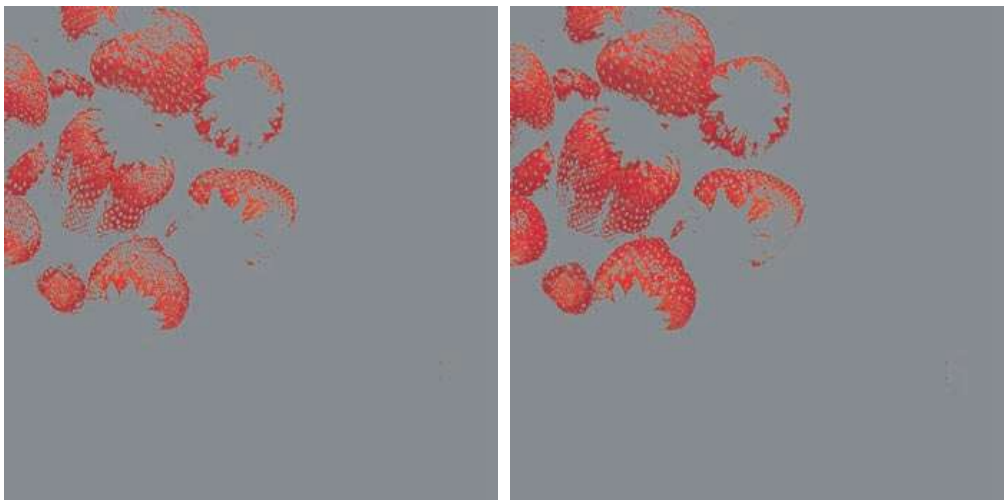
If a sphere is used to specify the colors of interest, Eq. (6-43) becomes

$$s_i = \begin{cases} 0.5 & \text{if } \sum_{j=1}^{n} (r_j - a_j)^2 > R_0^2 \\ r_i & \text{otherwise} \end{cases} \qquad i = 1, 2, \ldots, n \tag{6-44}$$

Here, R_0 is the radius of the enclosing sphere (or hypersphere for $n > 3$) and $(a_1, a_2, \ldots, a_n)$ are the components of its center (i.e., the prototypical color). Other useful variations of Eqs. (6-43) and (6-44) include implementing multiple color prototypes and reducing the intensity of the colors outside the region of interest—rather than setting them to a neutral constant.

EXAMPLE 6.8: Color slicing.

Equations (6-43) and (6-44) can be used to separate the strawberries in Fig. 6.29(a) from their sepals, cup, bowl, and other background elements. Figures 6.32(a) and (b) show the results of using both transformations. In each case, a prototype red with RGB color coordinate (0.6863, 0.1608, 0.1922) was selected from the most prominent strawberry. Parameters W and R_0 were chosen so that the highlighted region would not expand to other portions of the image. The actual values used, $W = 0.2549$ and $R_0 = 0.1765$, were determined interactively. Note that the sphere-based transformation of Eq. (6-44) performed slightly better, in the sense that it includes more of the strawberries' red areas. A sphere of radius 0.1765



a b

FIGURE 6.32 Color-slicing transformations that detect (a) reds within an RGB cube of width $W = 0.2549$ centered at $(0.6863, 0.1608, 0.1922)$, and (b) reds within an RGB sphere of radius 0.1765 centered at the same point. Pixels outside the cube and sphere were replaced by color $(0.5, 0.5, 0.5)$.

does not completely enclose a cube of width 0.2549, but it is not small enough to be completely enclosed by the cube either. In Section 6.7, and later in Chapter 10, you will learn more advanced techniques for using color and other multispectral information to extract objects from their background.

TONE AND COLOR CORRECTIONS

Problems involving an image's tonal range need to be corrected before color irregularities, such as over- and under-saturated colors, can be resolved, The tonal range of an image, also called its *key type*, refers to its general distribution of color intensities. Most of the information in *high-key* images is concentrated at high (or light) intensities; the colors of *low-key* images are located predominantly at low intensities; and *middle-key* images lie in between. As in the grayscale case, it is often desirable to distribute the intensities of a color image equally between the highlights and the shadows. The following examples illustrate a variety of color transformations for the correction of tonal and color imbalances.

EXAMPLE 6.9: Tonal transformations.

Transformations for modifying image tones normally are selected interactively. The idea is to adjust experimentally the image's brightness and contrast to provide maximum detail over a suitable range of intensities. The colors themselves are not changed. In the RGB and CMY(K) spaces, this means mapping all the color components, except K, with the same transformation function (see Fig. 6.29); in the HSI color space, only the intensity component is modified, as noted in the previous section.

Figure 6.33 shows typical RGB transformations used for correcting three common tonal imbalances—flat, light, and dark images. The S-shaped curve in the first row of the figure is ideal for boosting contrast

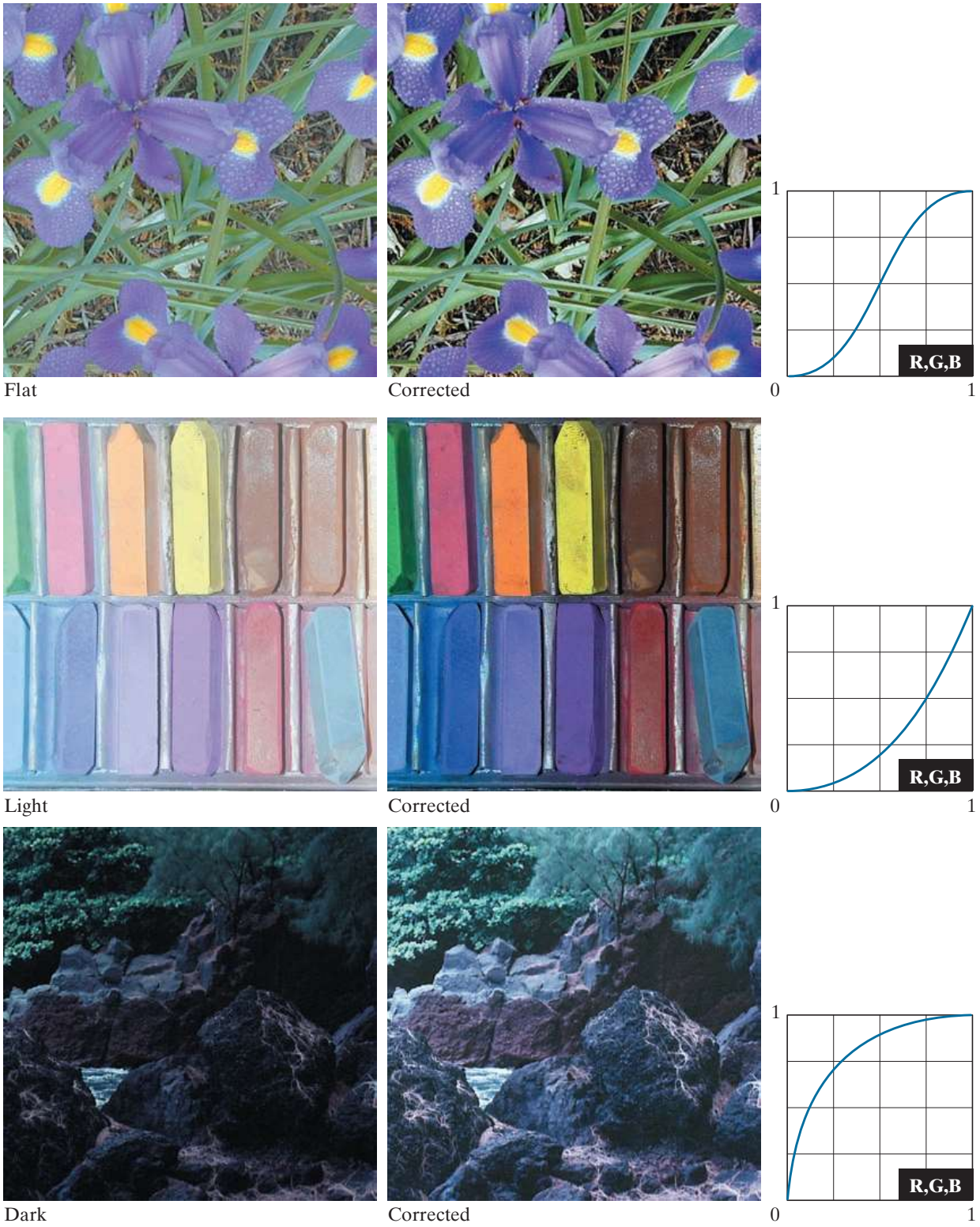


FIGURE 6.33 Tonal corrections for flat, light (high key), and dark (low key) color images. Adjusting the red, green, and blue components equally does not always alter the image hues significantly.

[see Fig. 3.2(a)]. Its midpoint is anchored so that highlight and shadow areas can be lightened and darkened, respectively. (The inverse of this curve can be used to correct excessive contrast.) The transformations in the second and third rows of the figure correct light and dark images, and are reminiscent of the power-law transformations in Fig. 3.6. Although the color components are discrete, as are the actual transformation functions, the transformation functions themselves are displayed and manipulated as continuous quantities—typically constructed from piecewise linear or higher order (for smoother mappings) polynomials. Note that the keys of the images in Fig. 6.33 are visually evident; they could also be determined using the histograms of the images' color components.

EXAMPLE 6.10: Color balancing.

Any color imbalances are addressed after the tonal characteristics of an image have been corrected. Although color imbalances can be determined directly by analyzing a known color in an image with a color spectrometer, accurate visual assessments are possible when white areas, where the RGB or CMY(K) components should be equal, are present. As Fig. 6.34 shows, skin tones are excellent subjects for visual color assessments because humans are highly perceptive of proper skin color. Vivid colors, such as bright red objects, are of little value when it comes to visual color assessment.

There are a variety of ways to correct color imbalances. When adjusting the color components of an image, it is important to realize that every action affects its overall color balance. That is, the perception of one color is affected by its surrounding colors. The color wheel of Fig. 6.30 can be used to predict how one color component will affect others. Based on the color wheel, for example, the proportion of any color can be increased by decreasing the amount of the opposite (or complementary) color in the image. Similarly, it can be increased by raising the proportion of the two immediately adjacent colors or decreasing the percentage of the two colors adjacent to the complement. Suppose, for instance, that there is too much magenta in an RGB image. It can be decreased: (1) by removing both red and blue, or (2) by adding green.

Figure 6.34 shows the transformations used to correct simple CMYK output imbalances. Note that the transformations depicted are the functions required for correcting the images; the inverses of these functions were used to generate the associated color imbalances. Together, the images are analogous to a color ring-around print of a darkroom environment and are useful as a reference tool for identifying color printing problems. Note, for example, that too much red can be due to excessive magenta (per the bottom left image) or too little cyan (as shown in the rightmost image of the second row).

HISTOGRAM PROCESSING OF COLOR IMAGES

Unlike the interactive enhancement approaches of the previous section, the gray-level histogram processing transformations of Section 3.3 can be applied to color images in an automated way. Recall that histogram equalization automatically determines a transformation that seeks to produce an image with a uniform histogram of intensity values. We showed in Section 3.3 that histogram processing can be quite successful at handling low-, high-, and middle-key images (for example, see Fig. 3.20). As you might suspect, it is generally unwise to histogram equalize the component images of a color image independently. This results in erroneous color. A more logical approach is to spread the color intensities uniformly, leaving the colors themselves (e.g., hues) unchanged. The following example shows that the HSI color space is ideally suited to this type of approach.

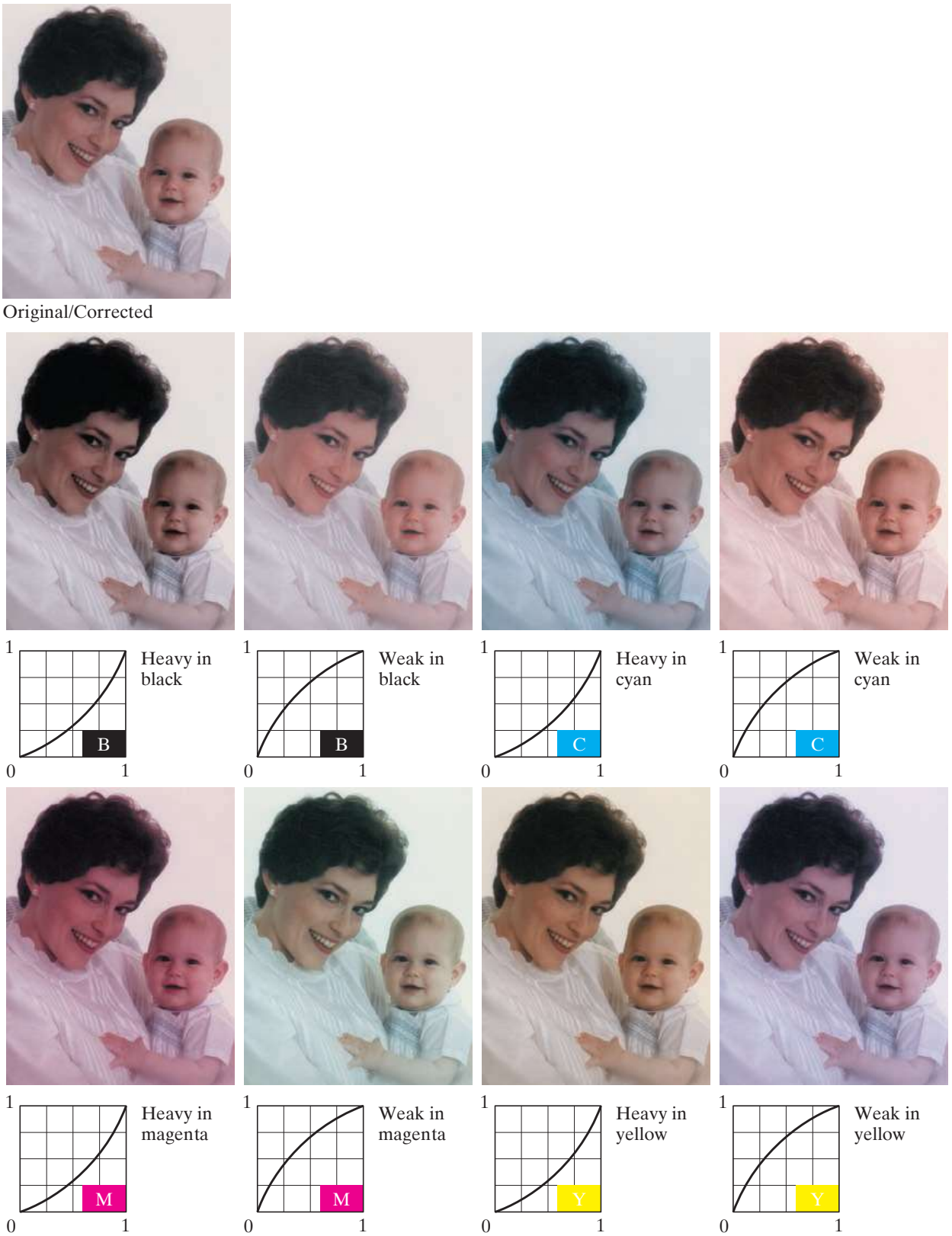


FIGURE 6.34 Color balancing a CMYK image.

a b
c d

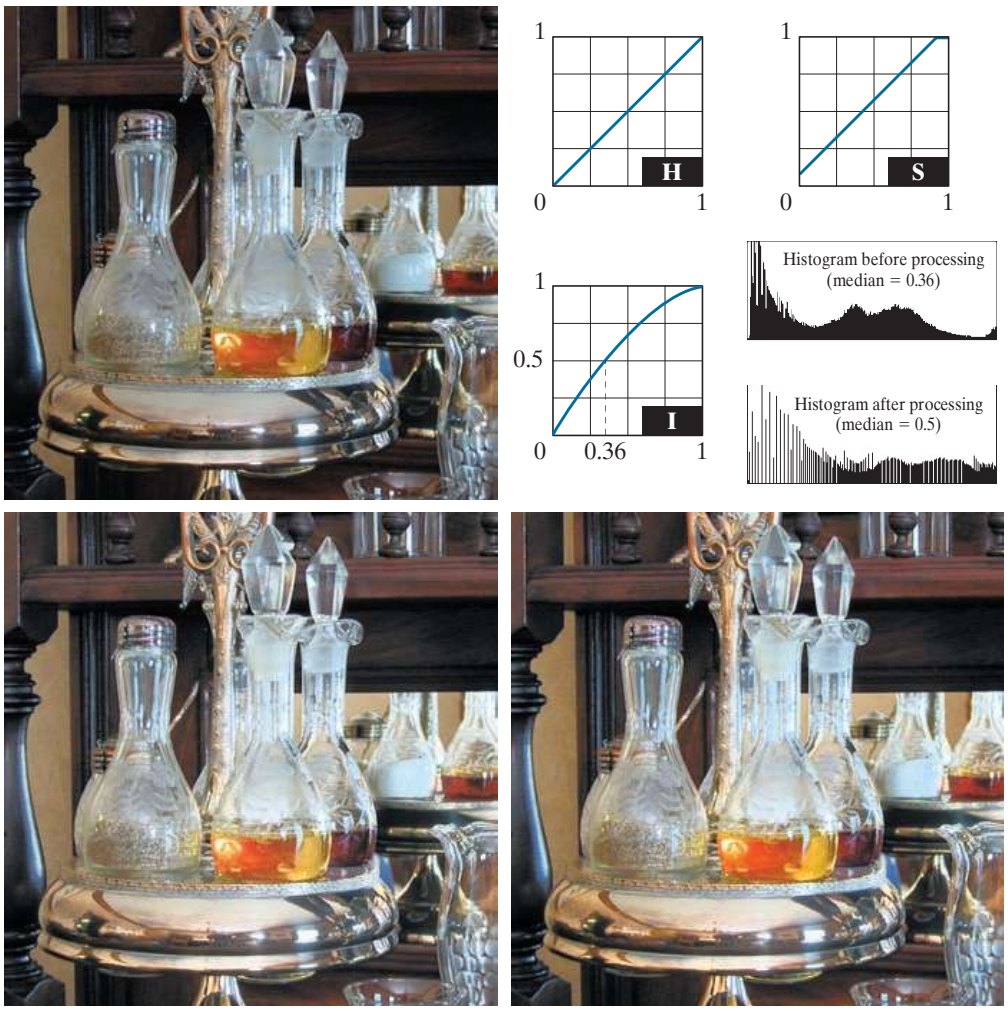


FIGURE 6.35 Histogram equalization (followed by saturation adjustment) in the HSI color space.

EXAMPLE 6.11: Histogram equalization in the HSI color space.

Figure 6.35(a) shows a color image of a caster stand containing cruets and shakers whose intensity component spans the entire (normalized) range of possible values, [0, 1]. As can be seen in the histogram of its intensity component prior to processing [see Fig. 6.35(b)], the image contains a large number of dark colors that reduce the median intensity to 0.36. Histogram equalizing the intensity component, without altering the hue and saturation, resulted in the image shown in Fig. 6.35(c). Note that the overall image is significantly brighter, and that several moldings and the grain of the wooden table on which the caster is sitting are now visible. Figure 6.35(b) shows the intensity histogram of the new image, as well as the intensity transformation used to equalize the intensity component [see Eq. (3-15)].

Although intensity equalization did not alter the values of hue and saturation of the image, it did impact the overall color perception. Note, in particular, the loss of vibrancy in the oil and vinegar in the cruets. Figure 6.35(d) shows the result of partially correcting this by increasing the image's saturation component, subsequent to histogram equalization, using the transformation in Fig. 6.35(b). This type of

adjustment is common when working with the intensity component in HSI space because changes in intensity usually affect the relative appearance of colors in an image.

6.6 COLOR IMAGE SMOOTHING AND SHARPENING

The next step beyond transforming each pixel of a color image without regard to its neighbors (as in the previous section) is to modify its value based on the characteristics of the surrounding pixels. In this section, the basics of this type of neighborhood processing will be illustrated within the context of color image smoothing and sharpening.

COLOR IMAGE SMOOTHING

With reference to Fig. 6.27(a) and the discussion in Sections 3.4 and 3.5, grayscale image smoothing can be viewed as a spatial filtering operation in which the coefficients of the filtering kernel have the same value. As the kernel is slid across the image to be smoothed, each pixel is replaced by the average of the pixels in the neighborhood encompassed by the kernel. As Fig. 6.27(b) shows, this concept is easily extended to the processing of full-color images. The principal difference is that instead of scalar intensity values, we must deal with component vectors of the form given in Eq. (6-37).

Let S_{xy} denote the set of coordinates defining a neighborhood centered at (x, y) in an RGB color image. The average of the RGB component vectors in this neighborhood is

$$\bar{\mathbf{c}}(x, y) = \frac{1}{K} \sum_{(s,t) \in S_{xy}} \mathbf{c}(s, t) \tag{6-45}$$

It follows from Eq. (6-37) and the properties of vector addition that

$$\bar{\mathbf{c}}(x, y) = \begin{bmatrix} \dfrac{1}{K} \displaystyle\sum_{(s,t) \in S_{xy}} R(s, t) \\ \dfrac{1}{K} \displaystyle\sum_{(s,t) \in S_{xy}} G(s, t) \\ \dfrac{1}{K} \displaystyle\sum_{(s,t) \in S_{xy}} B(s, t) \end{bmatrix} \tag{6-46}$$

We recognize the components of this vector as the scalar images that would be obtained by independently smoothing each plane of the original RGB image using conventional grayscale neighborhood processing. Thus, we conclude that smoothing by neighborhood averaging can be carried out on a per-color-plane basis. The result is the same as when the averaging is performed using RGB color vectors.

a b
c d

FIGURE 6.36
(a) RGB image. (b) Red component image. (c)Green component. (d) Blue component.

EXAMPLE 6.12: Color image smoothing by neighborhood averaging.

Consider the RGB color image in Fig. 6.36(a). Its three component images are shown in Figs. 6.36(b) through (d). Figures 6.37(a) through (c) show the HSI components of the image. Based on the discussion in the previous paragraph, we smoothed each component image of the RGB image in Fig. 6.36 independently using a 5×5 averaging kernel. We then combined the individually smoothed images to form the smoothed, full-color RGB result in Fig. 6.38(a). Note that this image appears as we would expect from performing a spatial smoothing operation, as in the examples given in Section 3.5.

In Section 6.2, we mentioned that an important advantage of the HSI color model is that it decouples intensity and color information. This makes it suitable for many grayscale processing techniques and suggests that it might be more efficient to smooth only the intensity component of the HSI representation in Fig. 6.37. To illustrate the merits and/or consequences of this approach, we next smooth only the intensity component (leaving the hue and saturation components unmodified) and convert the processed result to an RGB image for display. The smoothed color image is shown in Fig. 6.38(b). Note

a b c

FIGURE 6.37 HSI components of the RGB color image in Fig. 6.36(a). (a) Hue. (b) Saturation. (c) Intensity.

that it is similar to Fig. 6.38(a), but, as you can see from the difference image in Fig. 6.38(c), the two smoothed images are not identical. This is because in Fig. 6.38(a) the color of each pixel is the average color of the pixels in the neighborhood. On the other hand, by smoothing only the intensity component image in Fig. 6.38(b), the hue and saturation of each pixel was not affected and, therefore, the pixel colors did not change. It follows from this observation that the difference between the two smoothing approaches would become more pronounced as a function of increasing kernel size.

COLOR IMAGE SHARPENING

In this section we consider image sharpening using the Laplacian (see Section 3.6). From vector analysis, we know that the Laplacian of a vector is defined as a vector

a b c

FIGURE 6.38 Image smoothing with a 5×5 averaging kernel. (a) Result of processing each RGB component image. (b) Result of processing the intensity component of the HSI image and converting to RGB. (c) Difference between the two results.

whose components are equal to the Laplacian of the individual scalar components of the input vector. In the RGB color system, the Laplacian of vector $\mathbf{c}$ in Eq. (6-37) is

$$\nabla^2\left[\mathbf{c}(x,y)\right] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix} \tag{6-47}$$

which, as in the previous section, tells us that we can compute the Laplacian of a full-color image by computing the Laplacian of each component image separately.

EXAMPLE 6.13: Image sharpening using the Laplacian.

Figure 6.39(a) was obtained using Eq. (3-54) and the kernel in Fig. 3.45(c) to compute the Laplacians of the RGB component images in Fig. 6.36. These results were combined to produce the sharpened full-color result. Figure 6.39(b) shows a similarly sharpened image based on the HSI components in Fig. 6.37. This result was generated by combining the Laplacian of the intensity component with the unchanged hue and saturation components. The difference between the RGB and HSI sharpened images is shown in Fig. 6.39(c). The reason for the discrepancies between the two images is as in Example 6.12.

6.7 USING COLOR IN IMAGE SEGMENTATION

Segmentation is a process that partitions an image into regions. Although segmentation is the topic of Chapters 10 and 11, we consider color segmentation briefly here for the sake of continuity. You will have no difficulty following the discussion.

a b c

FIGURE 6.39 Image sharpening using the Laplacian. (a) Result of processing each RGB channel. (b) Result of processing the HSI intensity component and converting to RGB. (c) Difference between the two results.

SEGMENTATION IN HSI COLOR SPACE

If we wish to segment an image based on color and, in addition, we want to carry out the process on individual planes, it is natural to think first of the HSI space because color is conveniently represented in the hue image. Typically, saturation is used as a masking image in order to isolate further regions of interest in the hue image. The intensity image is used less frequently for segmentation of color images because it carries no color information. The following example is typical of how segmentation is performed in the HSI color space.

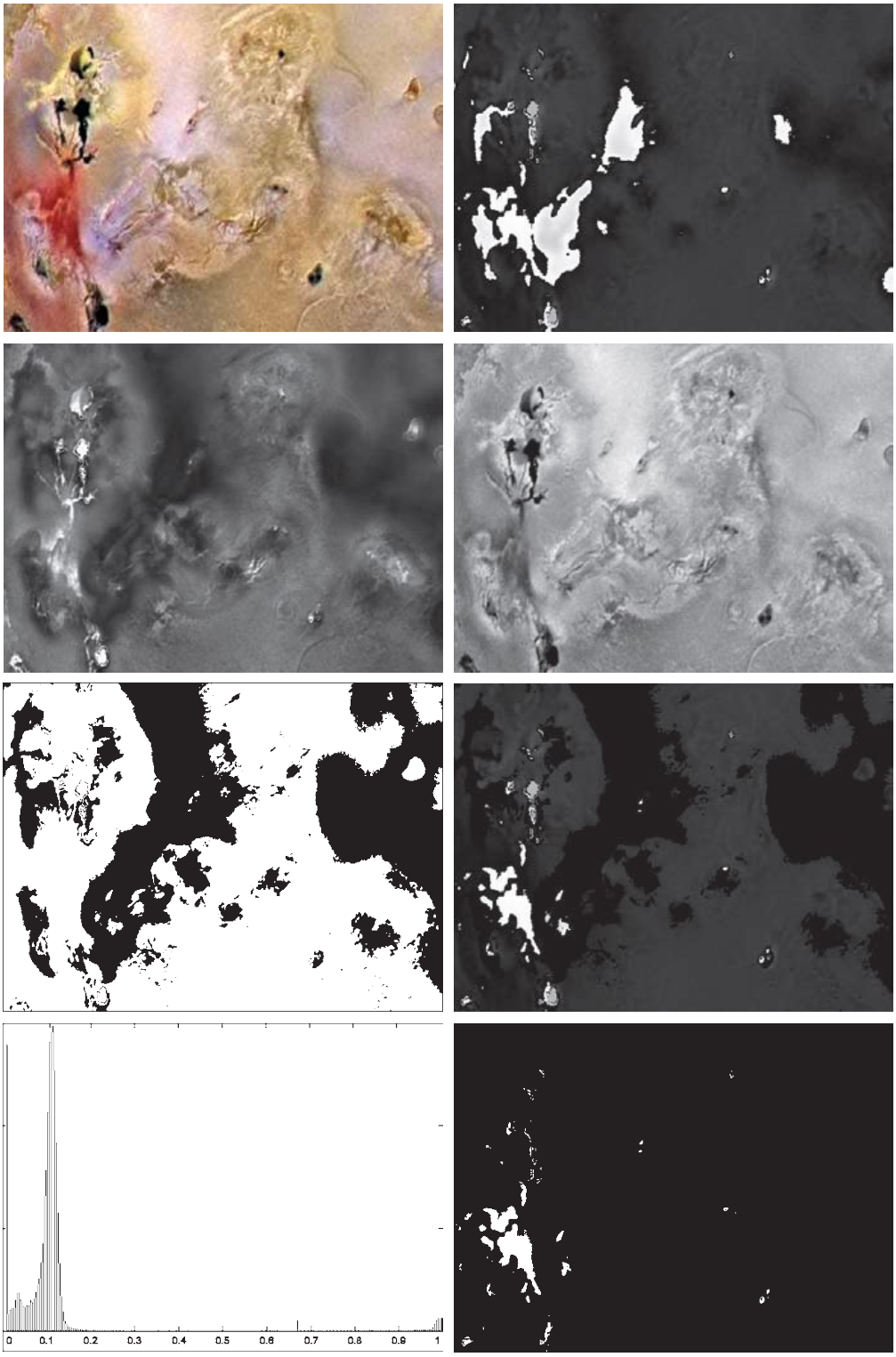
EXAMPLE 6.14: Segmenting a color image in HSI color space.

Suppose that it is of interest to segment the reddish region in the lower left of the image in Fig. 6.40(a). Figures 6.40(b) through (d) are its HSI component images. Note by comparing Figs. 6.40(a) and (b) that the region in which we are interested has relatively high values of hue, indicating that the colors are on the blue-magenta side of red (see Fig. 6.11). Figure 6.40(e) shows a binary mask generated by thresholding the saturation image with a threshold equal to 10% of the maximum value in that image. Any pixel value greater than the threshold was set to 1 (white). All others were set to 0 (black).

Figure 6.40(f) is the product of the mask with the hue image, and Fig. 6.40(g) is the histogram of the product image (note that the grayscale is in the range [0, 1]). We see in the histogram that high values (which are the values of interest) are grouped at the very high end of the grayscale, near 1.0. The result of thresholding the product image with threshold value of 0.9 resulted in the binary image in Fig. 6.40(h). The spatial location of the white points in this image identifies the points in the original image that have the reddish hue of interest. This was far from a perfect segmentation because there are points in the original image that we certainly would say have a reddish hue, but that were not identified by this segmentation method. However, it can be determined by experimentation that the regions shown in white in Fig. 6.40(h) are about the best this method can do in identifying the reddish components of the original image. The segmentation method discussed in the following section is capable of yielding better results.

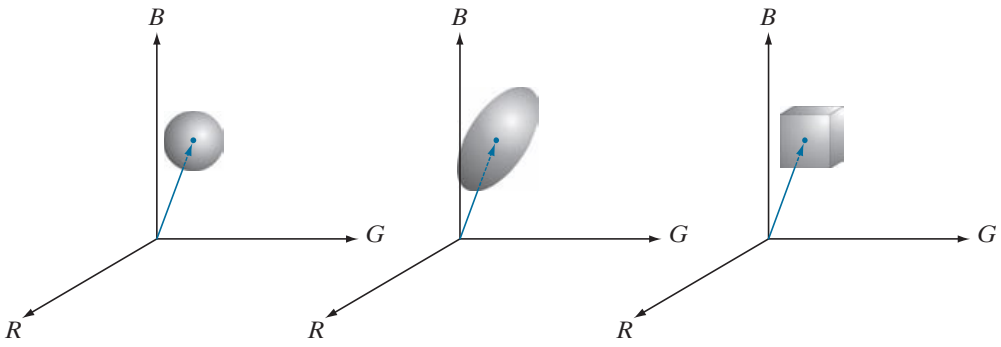
SEGMENTATION IN RGB SPACE

Although working in HSI space is more intuitive in the sense of colors being represented in a more familiar format, segmentation is one area in which better results generally are obtained by using RGB color vectors (see Fig. 6.7). The approach is straightforward. Suppose that the objective is to segment objects of a specified color range in an RGB image. Given a set of sample color points representative of the colors of interest, we obtain an estimate of the "average" color that we wish to segment. Let this average color be denoted by the RGB vector $\mathbf{a}$. The objective of segmentation is to classify each RGB pixel in a given image as having a color in the specified range or not. In order to perform this comparison, it is necessary to have a measure of similarity. One of the simplest measures is the Euclidean distance. Let $\mathbf{z}$ denote an arbitrary point in RGB space. We say that $\mathbf{z}$ is similar to $\mathbf{a}$ if the distance between them is less than a specified threshold, D_0. The Euclidean distance between $\mathbf{z}$ and $\mathbf{a}$ is given by



a b
c d
e f
g h

FIGURE 6.40 Image segmentation in HSI space. (a) Original. (b) Hue. (c) Saturation. (d) Intensity. (e) Binary saturation mask (black = 0). (f) Product of (b) and (e). (g) Histogram of (f). (h) Segmentation of red components from (a).



a b c

FIGURE 6.41 Three approaches for enclosing data regions for RGB vector segmentation.

$$D(\mathbf{z},\mathbf{a}) = \| \mathbf{z}-\mathbf{a} \| = \left[(\mathbf{z}-\mathbf{a})^T (\mathbf{z}-\mathbf{a}) \right]^{\frac{1}{2}} = \left[(\mathbf{z}_R-\mathbf{a}_R)^2 + (\mathbf{z}_G-\mathbf{a}_G)^2 + (\mathbf{z}_B-\mathbf{a}_B)^2 \right]^{\frac{1}{2}} \tag{6-48}$$

where the subscripts R, G, and B denote the RGB components of vectors **a** and **z**. The locus of points such that $D(\mathbf{z},\mathbf{a}) \le D_0$ is a solid sphere of radius D_0, as illustrated in Fig. 6.41(a). Points contained within the sphere satisfy the specified color criterion; points outside the sphere do not. Coding these two sets of points in the image with, say, black and white, produces a binary segmented image.

A useful generalization of Eq. (6-48) is a distance measure of the form

$$D(\mathbf{z},\mathbf{a}) = \left[(\mathbf{z}-\mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z}-\mathbf{a}) \right]^{\frac{1}{2}} \tag{6-49}$$

This equation is called the *Mahalanobis distance*. You are seeing it used here for *multivariate thresholding* (see Section 10.3 regarding thresholding).

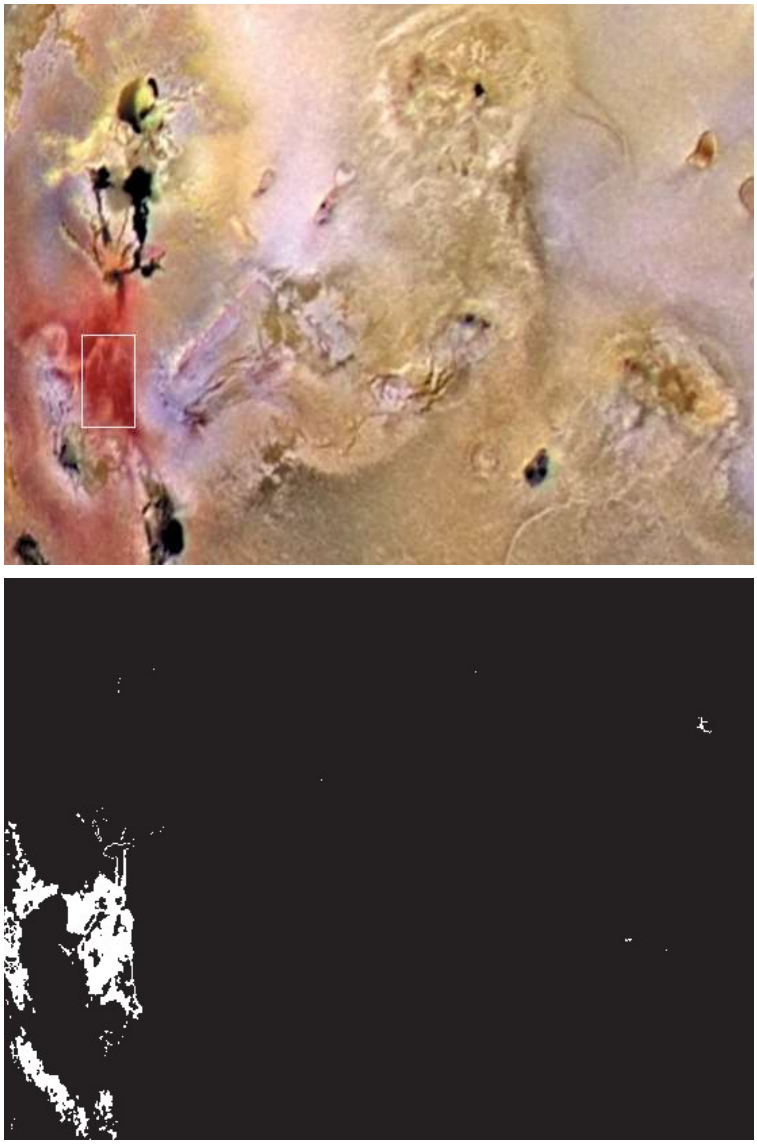
where **C** is the covariance matrix (see Section 11.5) of the samples chosen to be representative of the color range we wish to segment. The locus of points such that $D(\mathbf{z},\mathbf{a}) \le D_0$ describes a solid 3-D elliptical body [Fig. 6.41(b)] with the important property that its principal axes are oriented in the direction of maximum data spread. When $\mathbf{C} = \mathbf{I}$, the 3×3 identity matrix, Eq. (6-49) reduces to Eq. (6-48). Segmentation is as described in the preceding paragraph.

Because distances are positive and monotonic, we can work with the distance squared instead, thus avoiding square root computations. However, implementing Eq. (6-48) or (6-49) is computationally expensive for images of practical size, even if the square roots are not computed. A compromise is to use a bounding box, as illustrated in Fig. 6.41(c). In this approach, the box is centered on **a**, and its dimensions along each of the color axes is chosen proportional to the standard deviation of the samples along each of the axis. We use the sample data to compute the standard deviations, which are the parameters used for segmentation with this approach. Given an arbitrary color point, we segment it by determining whether or not it is on the surface or inside the box, as with the distance formulations. However, determining whether a color point is inside or outside a box is much simpler computationally

when compared to a spherical or elliptical enclosure. Note that the preceding discussion is a generalization of the color-slicing method introduced in Section 6.5.

EXAMPLE 6.15: Color segmentation in RGB color space.

The rectangular region shown Fig. 6.42(a) contains samples of reddish colors we wish to segment out of the color image. This is the same problem we considered in Example 6.14 using hue, but now we approach the problem using RGB color vectors. The approach followed was to compute the mean vector **a** using the color points contained within the rectangle in Fig. 6.42(a), and then to compute the standard deviation of the red, green, and blue values of those samples. A box was centered at **a**, and its dimensions along each of the RGB axes were selected as 1.25 times the standard deviation of the data along the corresponding axis. For example, let σ_R denote the standard deviation of the red components



a
b

FIGURE 6.42
Segmentation in RGB space. (a) Original image with colors of interest shown enclosed by a rectangle. (b) Result of segmentation in RGB vector space. Compare with Fig. 6.40(h).

of the sample points. Then the dimensions of the box along the R-axis extended from $(a_R - 1.25\sigma_R)$ to $(a_R + 1.25\sigma_R)$, where a_R is the red component of average vector **a**. Figure 6.42(b) shows the result of coding each point in the color image as white if it was on the surface or inside the box, and as black otherwise. Note how the segmented region was generalized from the color samples enclosed by the rectangle. In fact, by comparing Figs. 6.42(b) and 6.40(h), we see that segmentation in the RGB vector space yielded results that are much more accurate, in the sense that they correspond much more closely with what we would define as "reddish" points in the original color image. This result is not unexpected, because in the RGB space we used three color variables, as opposed to just one in the HSI space.

COLOR EDGE DETECTION

As we will discuss in Section 10.2, edge detection is an important tool for image segmentation. In this section, we are interested in the issue of computing edges on individual component images, as opposed to computing edges directly in color vector space.

We introduced edge detection by gradient operators in Section 3.6, when discussing image sharpening. Unfortunately, the gradient discussed there is not defined for vector quantities. Thus, we know immediately that computing the gradient on individual images and then using the results to form a color image will lead to erroneous results. A simple example will help illustrate the reason why.

Consider the two $M \times M$ color images (M odd) in Figs. 6.43(d) and (h), composed of the three component images in Figs. 6.43(a) through (c) and (e) through (g), respectively. If, for example, we compute the gradient image of each of the component images using Eq. (3-58), then add the results to form the two corresponding RGB gradient images, the value of the gradient at point $[(M+1)/2, (M+1)/2]$ would be the same in both cases. Intuitively, we would expect the gradient at that point to be stronger for the image in Fig. 6.43(d) because the edges of the R, G, and B images are in the same direction in that image, as opposed to the image in Fig. 6.43(h), in which only two of the edges are in the same direction. Thus we see from this simple example that processing the three individual planes to form a composite gradient image can yield erroneous results. If the problem is one of just detecting edges, then the individual-component approach can yield acceptable results. If accuracy is an issue, however, then obviously we need a new definition of the gradient applicable to vector quantities. We discuss next a method proposed by Di Zenzo [1986] for doing this.

The problem at hand is to define the gradient (magnitude and direction) of the vector **c** in Eq. (6-37) at any point (x, y). As we just mentioned, the gradient we studied in Section 3.6 is applicable to a *scalar* function $f(x, y)$; it is not applicable to vector functions. The following is one of the various ways in which we can extend the concept of a gradient to vector functions. Recall that for a scalar function $f(x, y)$, the gradient is a vector pointing in the direction of maximum rate of change of f at coordinates (x, y).

Let **r**, **g**, and **b** be unit vectors along the R, G, and B axis of RGB color space (see Fig. 6.7), and define the vectors

a b c d
e f g h

FIGURE 6.43 (a)–(c) R, G, and B component images, and (d) resulting RGB color image. (e)–(g) R, G, and B component images, and (h) resulting RGB color image.

$$\mathbf{u} = \frac{\partial R}{\partial x}\mathbf{r} + \frac{\partial G}{\partial x}\mathbf{g} + \frac{\partial B}{\partial x}\mathbf{b} \tag{6-50}$$

and

$$\mathbf{v} = \frac{\partial R}{\partial y}\mathbf{r} + \frac{\partial G}{\partial y}\mathbf{g} + \frac{\partial B}{\partial y}\mathbf{b} \tag{6-51}$$

Let the quantities g_{xx}, g_{yy}, and g_{xy} be defined in terms of the dot product of these vectors, as follows:

$$g_{xx} = \mathbf{u} \cdot \mathbf{u} = \mathbf{u}^T\mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2 \tag{6-52}$$

$$g_{yy} = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^T\mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2 \tag{6-53}$$

and

$$g_{xy} = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T\mathbf{v} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y} \tag{6-54}$$

Keep in mind that R, G, and B, and consequently the g's, are functions of x and y. Using this notation, it can be shown (Di Zenzo [1986]) that the direction of maximum rate of change of $\mathbf{c}(x, y)$ is given by the angle

$$\theta(x,y) = \frac{1}{2}\tan^{-1}\left[\frac{2g_{xy}}{g_{xx} - g_{yy}}\right] \tag{6-55}$$

and that the value of the rate of change at (x, y) in the direction of $\theta(x, y)$ is given by

$$F_\theta(x,y) = \left\{\frac{1}{2}\left[\left(g_{xx} + g_{yy}\right) + \left(g_{xx} - g_{yy}\right)\cos 2\theta(x,y) + 2g_{xy}\sin 2\theta(x,y)\right]\right\}^{\frac{1}{2}} \tag{6-56}$$

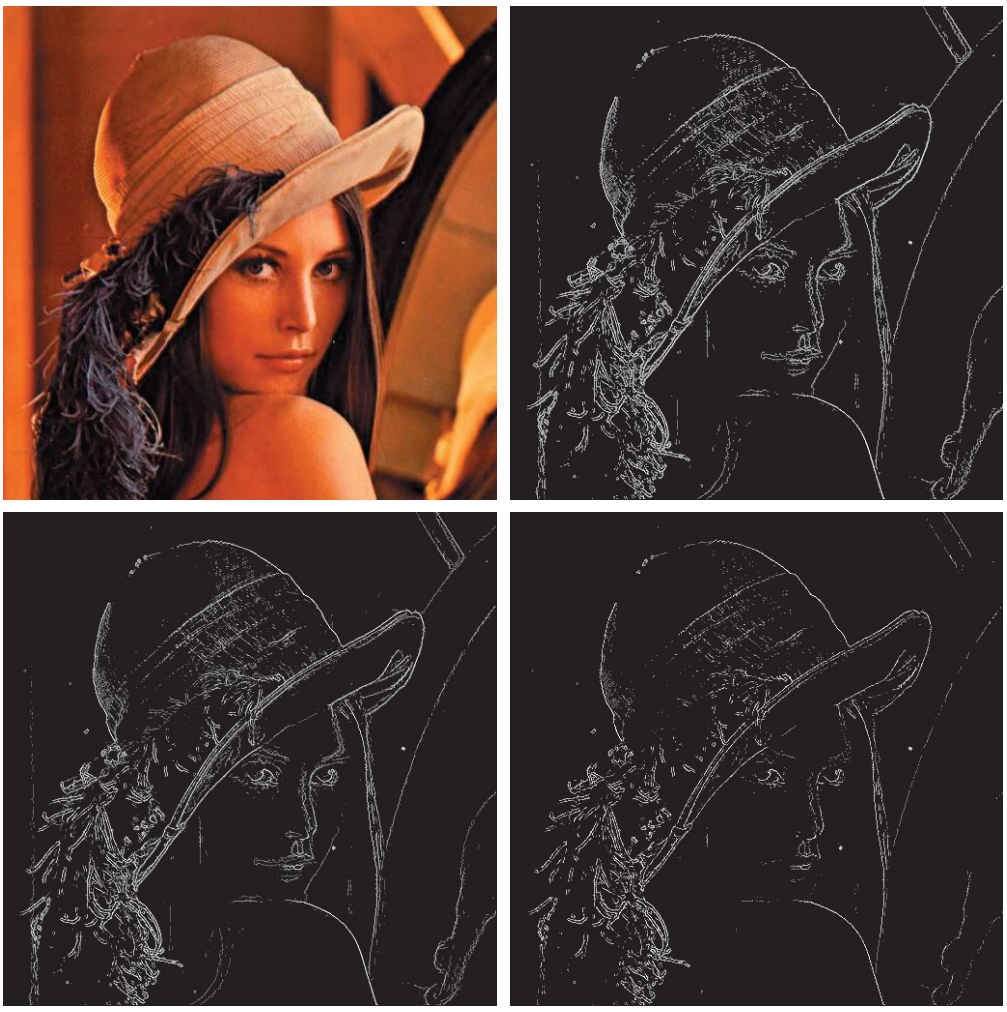
Because $\tan(\alpha) = \tan(\alpha \pm \pi)$, if θ_0 is a solution to Eq. (6-55), so is $\theta_0 \pm \pi/2$. Furthermore, $F_\theta = F_{\theta+\pi}$, so F has to be computed only for values of θ in the half-open interval $[0, \pi)$. The fact that Eq. (6-55) gives two values 90° apart means that this equation associates with each point (x, y) a pair of orthogonal directions. Along one of those directions F is maximum, and it is minimum along the other. The derivation of these results is rather lengthy, and we would gain little in terms of the fundamental objective of our current discussion by detailing it here. Consult the paper by Di Zenzo [1986] for details. The Sobel operators discussed in Section 3.6 can be used to compute the partial derivatives required for implementing Eqs. (6-52) through (6-54).

EXAMPLE 6.16: Edge detection in RGB vector space.

Figure 6.44(b) is the gradient of the image in Fig. 6.44(a), obtained using the vector method just discussed. Figure 6.44(c) shows the image obtained by computing the gradient of each RGB component image and forming a composite gradient image by adding the corresponding values of the three component images at each coordinate (x, y). The edge detail of the vector gradient image is more complete than the detail in the individual-plane gradient image in Fig. 6.44(c); for example, see the detail around the subject's right eye. The image in Fig. 6.44(d) shows the difference between the two gradient images at each point (x, y). It is important to note that both approaches yielded reasonable results. Whether the extra detail in Fig. 6.44(b) is worth the added computational burden over the Sobel operator computations can only be determined by the requirements of a given problem. Figure 6.45 shows the three component gradient images, which, when added and scaled, were used to obtain Fig. 6.44(c).

6.8 NOISE IN COLOR IMAGES

The noise models discussed in Section 5.2 are applicable to color images. Usually, the noise content of a color image has the same characteristics in each color channel, but it is possible for color channels to be affected differently by noise. One possibility is for the electronics of a particular channel to malfunction. However, different noise levels are more likely caused by differences in the relative strength of illumination available to each of the color channels. For example, use of a red filter in a CCD camera will reduce the strength of illumination detected by the red sensing elements. CCD sensors are noisier at lower levels of illumination, so the resulting red com-



a b
c d

FIGURE 6.44
(a) RGB image. (b) Gradient computed in RGB color vector space. (c) Gradient image formed by the elementwise sum of three individual gradient images, each computed using the Sobel operators. (d) Difference between (b) and (c).

a b c

FIGURE 6.45 Component gradient images of the color image in Fig. 6.44. (a) Red component, (b) green component, and (c) blue component. These three images were added and scaled to produce the image in Fig. 6.44(c).

ponent of an RGB image would tend to be noisier than the other two component images in this situation.

EXAMPLE 6.17: Illustration of the effects of noise when converting noisy RGB images to HSI.

In this example, we take a brief look at noise in color images and how noise carries over when converting from one color model to another. Figures 6.46(a) through (c) show the three color planes of an RGB image corrupted by additive Gaussian noise, and Fig. 6.46(d) is the composite RGB image. Note that fine grain noise such as this tends to be less visually noticeable in a color image than it is in a grayscale image. Figures 6.47(a) through (c) show the result of converting the RGB image in Fig. 6.46(d) to HSI. Compare these results with the HSI components of the original image (see Fig. 6.37) and note how significantly degraded the hue and saturation components of the noisy image are. This was caused by the nonlinearity of the cos and min operations in Eqs. (6-17) and (6-18), respectively. On the other hand, the intensity component in Fig. 6.47(c) is slightly smoother than any of the three noisy RGB component images. This is because the intensity image is the average of the RGB images, as indicated in Eq. (6-19). (Recall the discussion in Section 2.6 regarding the fact that image averaging reduces random noise.)

a b
c d

FIGURE 6.46 (a)–(c) Red, green, and blue 8-bit component images corrupted by additive Gaussian noise of mean 0 and standard deviation of 28 intensity levels. (d) Resulting RGB image. [Compare (d) with Fig. 6.44(a).]

a b c

FIGURE 6.47 HSI components of the noisy color image in Fig. 6.46(d). (a) Hue. (b) Saturation. (c) Intensity.

In cases when, say, only one RGB channel is affected by noise, conversion to HSI spreads the noise to all HSI component images. Figure 6.48 shows an example. Figure 6.48(a) shows an RGB image whose green component image is corrupted by salt-and-pepper noise, with a probability of either salt or pepper equal to 0.05. The HSI component images in Figs. 6.48(b) through (d) show clearly how the noise spread from the green RGB channel to all the HSI images. Of course, this is not unexpected because computation of the HSI components makes use of all RGB components, as discussed in Section 6.2.

As is true of the processes we have discussed thus far, filtering of full-color images can be carried out on a per-image basis, or directly in color vector space, depending on the process. For example, noise reduction by using an averaging filter is the process discussed in Section 6.6, which we know gives the same result in vector space as it does if the component images are processed independently. However, other filters cannot be formulated in this manner. Examples include the class of order statistics filters discussed in Section 5.3. For instance, to implement a median filter in color vector space it is necessary to find a scheme for ordering vectors in a way that the median makes sense. While this was a simple process when dealing with scalars, the process is considerably more complex when dealing with vectors. A discussion of vector ordering is beyond the scope of our discussion here, but the book by Plataniotis and Venetsanopoulos [2000] is a good reference on vector ordering and some of the filters based on the concept of ordering.

6.9 COLOR IMAGE COMPRESSION

Because the number of bits required to represent color is typically three to four times greater than the number employed in the representation of gray levels, data compression plays a central role in the storage and transmission of color images. With respect to the RGB, CMY(K), and HSI images of the previous sections, the data that are the object of any compression are the components of each color pixel (e.g., the red, green, and blue components of the pixels in an RGB image); they are

a b
c d

FIGURE 6.48
(a) RGB image with green plane corrupted by salt-and-pepper noise. (b) Hue component of HSI image. (c) Saturation component. (d) Intensity component.

the means by which the color information is conveyed. Compression is the process of reducing or eliminating redundant and/or irrelevant data. Although compression is the topic of Chapter 8, we illustrate the concept briefly in the following example using a color image.

EXAMPLE 6.18: An example of color image compression.

Figure 6.49(a) shows a 24-bit RGB full-color image of an iris, in which 8 bits each are used to represent the red, green, and blue components. Figure 6.49(b) was reconstructed from a compressed version of the image in (a) and is, in fact, a compressed and subsequently decompressed approximation of it. Although the compressed image is not directly displayable—it must be decompressed before input to a color monitor—the compressed image contains only 1 data bit (and thus 1 storage bit) for every 230 bits of data in the original image (you will learn about the origin of these numbers in Chapter 8). Suppose that the image is of size $2000 \times 3000 = 6 \cdot 10^6$ pixels. The image is 24 bits/pixel, so it storage size is $144 \cdot 10^6$ bits.

a
b

FIGURE 6.49
Color image compression. (a) Original RGB image. (b) Result of compressing, then decompressing the image in (a).

Suppose that you are sitting at an airport waiting for your flight, and want to upload 100 such images using the airport's public WiFi connection. At a (relatively high) upload speed of $10 \cdot 10^6$ bits/sec, it would take you about 24 min to upload your images. In contrast, the compressed images would take about 6 sec to upload. Of course, the transmitted data would have to be decompressed at the other end for viewing, but the decompression can be done in a matter of seconds. Note that the reconstructed approximation image is slightly blurred. This is a characteristic of many lossy compression techniques; it can be reduced or eliminated by changing the level of compression. The JPEG 2000 compression algorithm used to generate Fig. 6.49(b) is described in detail in Section 8.2.

Summary, References, and Further Reading

The material in this chapter is an introduction to color image processing and covers topics selected to provide a solid background in the techniques used in this branch of image processing. Our treatment of color fundamentals and color models was prepared as foundation material for a field that is wide in technical scope and areas of application. In particular, we focused on color models that we felt are not only useful in digital image processing but provide also the tools necessary for further study in this area of image processing. The discussion of pseudocolor and full-color processing on an individual image basis provides a tie to techniques that were covered in some detail in Chapters 3 through 5. The material on color vector spaces is a departure from methods that we had studied before and highlights some important differences between grayscale and full-color processing. Our treatment of noise in color images also points out that the vector nature of the problem, along with the fact that color images are routinely transformed from one working space to another, has implications on the issue of how to reduce noise in these images. In some cases, noise filtering can be done on a per-image basis, but others, such as median filtering, require special treatment to reflect the fact that color pixels are vector quantities, as mentioned earlier. Although segmentation is the topic of Chapters 10 and 11, and image data compression is the topic of Chapter 8, we introduced them briefly in the context of color image processing.

For a comprehensive reference on the science of color, see Malacara [2011]. Regarding the physiology of color, see Snowden et al. [2012]. These two references, together with the book by Kuehni [2012], provide ample supplementary material for the discussion in Section 6.1. For further reading on color models (Section 6.2), see Fortner and Meyer [1997], Poynton [1996], and Fairchild [1998]. For a detailed derivation of the equations for the HSI model see the paper by Smith [1978] or consult the book website. The topic of pseudocolor (Section 6.3) is closely tied to the general area of image data visualization. Wolff and Yaeger [1993] is a good basic reference on the use of pseudocolor. See also Telea [2008]. For additional reading on the material in Sections 6.4 and 6.5, see Plataniotis and Venetsanopoulos [2000]. The material on color image filtering (Section 6.6) is based on the vector formulation introduced in Section 6.4 and on our discussion of spatial filtering in Chapter 3. The area of color image segmentation (Section 6.7) is of significant current interest. For an overview of current trends in this field see the survey by Vantaram and Saber [2012]. For more advanced color image processing techniques than those discussed in this chapter see Fernandez-Maloigne [2012]. The discussion in Section 6.8 is based on the noise models introduced in Section 5.2. References on color image compression (Section 6.9) are listed at the end of Chapter 8. For details of software implementation of many of the techniques discussed 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).

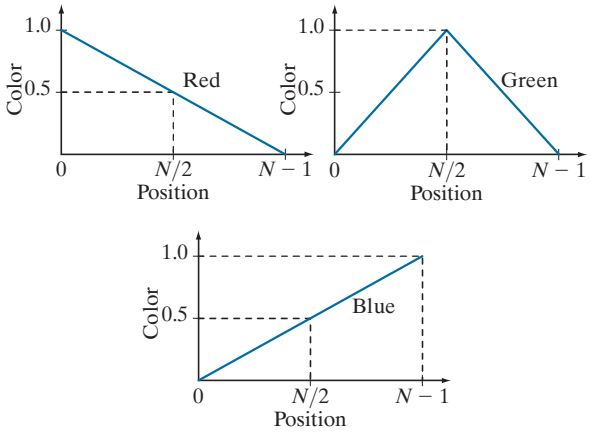
6.1 Give the percentages of red (X), green (Y), and blue (Z) light required to generate the point labeled "warm white" in Fig. 6.5.

6.2* Consider any two valid colors c_1 and c_2 with coordinates (x_1, y_1) and (x_2, y_2) in the chromaticity diagram of Fig. 6.5. Derive the necessary general expression(s) for computing the relative percentages of colors c_1 and c_2 composing any color that is known to lie on the straight line joining these two colors.

6.3 Consider any three valid colors c_1, c_2, and c_3 with coordinates (x_1, y_1), (x_2, y_2), and (x_3, y_3), in the chromaticity diagram of Fig. 6.5. Derive the necessary general expression(s) for computing the relative percentages of c_1, c_2, and c_3 composing a color that is known to lie within the triangle whose vertices are at the coordinates of c_1, c_2, and c_3.

6.4* In an automated assembly application, three types of parts are to be color-coded to simplify detection. However, only a monochrome TV camera is available to acquire digital images. Propose a technique for using this camera to detect the three different colors.

6.5 The R, G, and B component images of an RGB image have the horizontal intensity profiles shown in the following diagram. What color would a person see in the middle column of this image?



6.6* Sketch the RGB components of the following image as they would appear on a monochrome monitor. All colors are at maximum intensity and saturation. In working this problem, consider the gray border as part of the image.

6.7 What is the maximum number of possible different shades of gray in an RGB image whose three component images are 8-bit images?

6.8 Consider the RGB cube in Fig. 6.8 and answer each of the following questions.

(a)* Describe how the gray levels vary in each of the R, G, and B primary images that make up the front face of the color cube (this is the face closer to you). Assume that each component image is an 8-bit image.

(b) Suppose that we replace every color in the RGB cube by its CMY color. This new cube is displayed on an RGB monitor. Label with a color name the eight vertices of the new cube that you would see on the screen.

(c) What can you say about the colors on the edges of the RGB color cube regarding saturation?

6.9 Do the following.

(a)* Sketch the CMY components of the image in Problem 6.6 as they would appear on a monochrome monitor.

(b) If the CMY components sketched in (a) are fed into the red, green, and blue inputs of a color monitor, respectively, describe the appearance of the resulting image.

6.10* Sketch the HSI components of the image in Problem 6.6 as they would appear on a monochrome monitor.

6.11 Propose a method for generating a color band similar to the one shown in the zoomed section entitled *Visible Spectrum* in Fig. 6.2. Note that the band starts at a dark purple on the left and proceeds toward pure red on the right. (*Hint:* Use the HSI color model.)

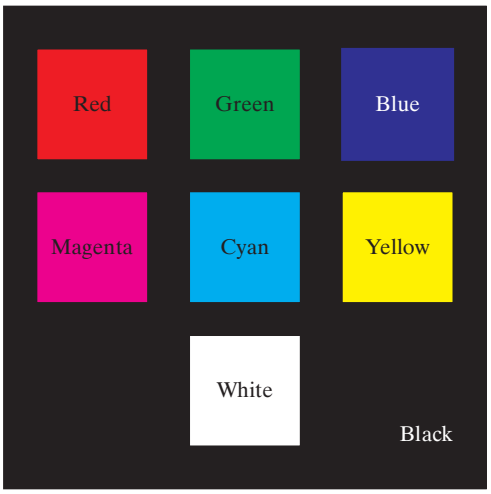
6.12* Propose a method for generating a color version of the image shown diagrammatically in Fig. 6.11(c). Give your answer in the form of a flow chart. Assume that the intensity value is fixed and given. (*Hint:* Use the HSI color model.)

6.13 Consider the following image composed of solid color squares. For discussing your answer, choose a gray scale consisting of eight shades of gray, 0 through 7, where 0 is black and 7 is white. Suppose that the image is converted to HSI color space. In answering the following questions, use specific numbers for the gray shades if using numbers makes sense. Otherwise, the relationships "same as," "lighter than," or "darker than" are sufficient. If you cannot assign a specific gray level or one of these relationships to the image you are discussing, give the reason.

(a)* Sketch the hue image.

(b) Sketch the saturation image.

(c) Sketch the intensity image.

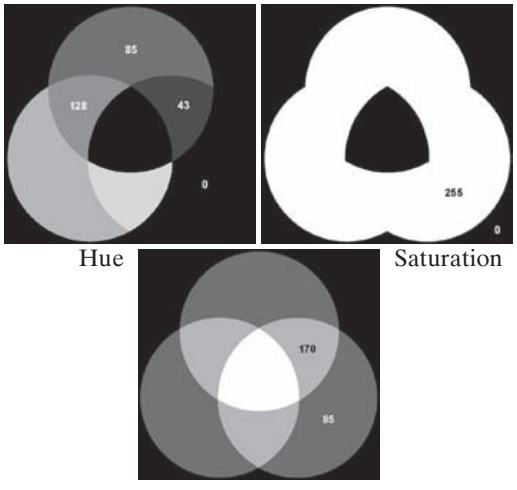


6.14 The following 8-bit images are the H, S, and I component images from Fig. 6.14. The numbers indicate gray-level values. Answer the following questions, explaining the basis for your answer in each. If it is not possible to answer a question based on the given information, state why you cannot do so.

(a)* Give the gray-level values of all regions in the hue image.

(b) Give the gray-level value of all regions in the saturation image.

(c) Give the gray-level values of all regions in the intensity image.



Hue Saturation

Intensity

6.15* Compute the $L^*a^*b^*$ components of the image in Problem 6.6 assuming:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.588 & 0.179 & 0.183 \\ 0.29 & 0.606 & 0.105 \\ 0 & 0.068 & 1.021 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This matrix equation defines the tristimulus values of the colors generated by the standard National Television System Committee (NTSC) color TV phosphors viewed under *D65* standard illumination (Benson [1985]).

6.16* Derive the CMY intensity mapping function of Eq. (6-41) from the RGB counterpart in Eq. (6-40). [*Hint:* Start with Eq. (6-5).]

6.17 Start with Eqs. (6-6)-(6-12) and derive Eq. (6-42). (*Hint:* The intensity of the CMYK image is changed by changing the K component only.)

6.18 Refer to Fig. 6.25 in answering the following:

(a)* Why does the image in Fig. 6.25(e) exhibit predominantly red tones?

(b)* Suggest an automated procedure for coding the water in Fig. 6.25 in a bright-blue color.

(c) Suggest an automated procedure for coding the predominantly man-made components in a bright yellow color. [*Hint:* Work with Fig. 6.25(e).]

6.19* Show that the saturation component of the complement of a color image cannot be computed from the saturation component of the input image alone.

6.20 Explain the shape of the hue transformation function for the image complement approximation in Fig. 6.31(b) using the HSI color model.

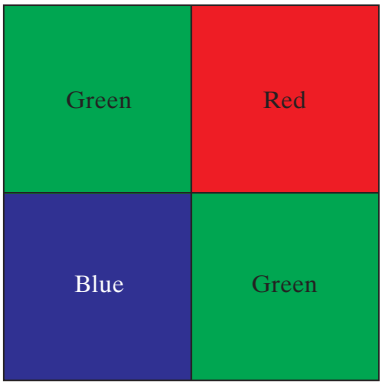
6.21* Derive the CMY transformations to generate the complement of a color image.

6.22 Draw the general shape of the transformation functions used to correct excessive contrast in the RGB color space.

6.23* Assume that the monitor and printer of an imaging system are imperfectly calibrated. An image that looks balanced on the monitor appears yellowish in print. Describe general transformations that might correct the imbalance. (*Hints:* Refer to the color wheel in Fig. 6.30 and the discussion of the $L^*a^*b^*$ color system in Section 6.2.)

6.24* Given an image in the RGB, CMY, or CMYK color system, how would you implement the color equivalent of gray-scale histogram matching (specification) from Section 3.3?

6.25 Consider the following 500×500 RGB image, in which the squares are fully saturated red, green, and blue, and each of the colors is at maximum intensity. An HSI image is generated from this image. Answer the following questions.



(a) Describe the appearance of each HSI component image.

(b)* The saturation component of the HSI image is smoothed using an averaging kernel of size 125×125. Describe the appearance of the result. (You may ignore image border effects in the filtering operation.)

(c) Repeat (b) for the hue image.

6.26 Answer the following.

(a)* Refer to the discussion in Section 6.7 about segmentation in the RGB color space. Give a procedure (in flow chart form) for determining whether a color vector (point) $\mathbf{z}$ is inside a cube with sides W, centered at an average color vector $\mathbf{a}$. Distance computations are not allowed.

(b) If the box is aligned with the axes this process also can be implemented on an image-by-image basis. Show how you would do it.

6.27 Show that Eq. (6-49) reduces to Eq. (6-48) when $\mathbf{C} = \mathbf{I}$, the identity matrix.

6.28 Sketch the surface in RGB space for the points that satisfy the equation

$$D(\mathbf{z},\mathbf{a}) = \left[(\mathbf{z}-\mathbf{a})^T \mathbf{C}^{-1} (\mathbf{z}-\mathbf{a})\right]^{\frac{1}{2}} = D_0$$

where D_0 is a positive constant. Assume that $\mathbf{a} = \mathbf{0}$, and that

$$\mathbf{C} = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.29 Refer to the discussion on color edge detection in Section 6.7. One might think that a logical approach for defining the gradient of an RGB image at any point (x, y) would be to compute the gradient vector (see Section 3.6) of each component image and then form a gradient vector for the color image by summing the three individual gradient vectors. Unfortunately, this method can at times yield erroneous results. Specifically, it is possible for a color image with clearly defined edges to have a zero gradient if this method were used. Give an example of such an image. (*Hint:* To simplify your analysis, set one of the color planes to a constant value.)

This page intentionally left blank