Due to these absorption characteristics, colors are seen as variable combinations of so called "**primary colors**" red, green and blue.

In **1931**, CIE (International Commission on Illumitation) designated the following:
Blue = 435.8nm
Green = 546.1nm
Red = 700nm

The shown detailed experimental curves became available 1965.

- Remember that there is no single color called red, green or blue in the color spectrum!
- Also, these fixed RGB components cannot generate ALL spectrum colors!

Color Image Processing: Color Image Representation

Colors are distinguished from another by **brightness**, **hue**, and **saturation**.

Brightness embodies the achromatic notion of intensity.

Hue is associated with the dominant wavelength in the mixture and represents the perceived dominant color.

Saturation refers to the relative purity or the amount of white light mixed with a hue.

Together hue and saturation are called **chromaticity**.

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

A color is specified by its trichromatic coefficients defined as:

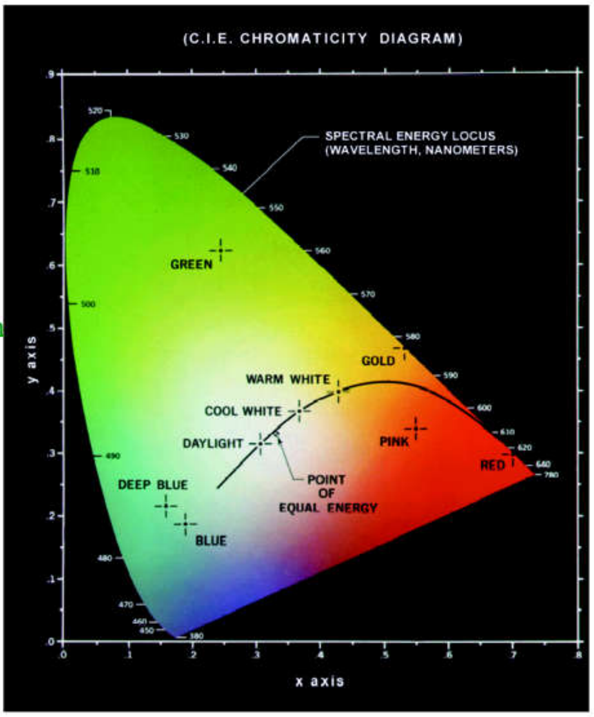
$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z}$$

Color Image Processing

FIGURE 6.5 Chromaticity diagram. (Courtesy of the General Electric Co., Lamp Business Division.)



- Pure colors are mapped on the boundary of the chromaticity diagram, fully saturated colors
- Colors inside the diagram are combinations of these colors
- Reference white is the point of equal energy, with zero saturation value

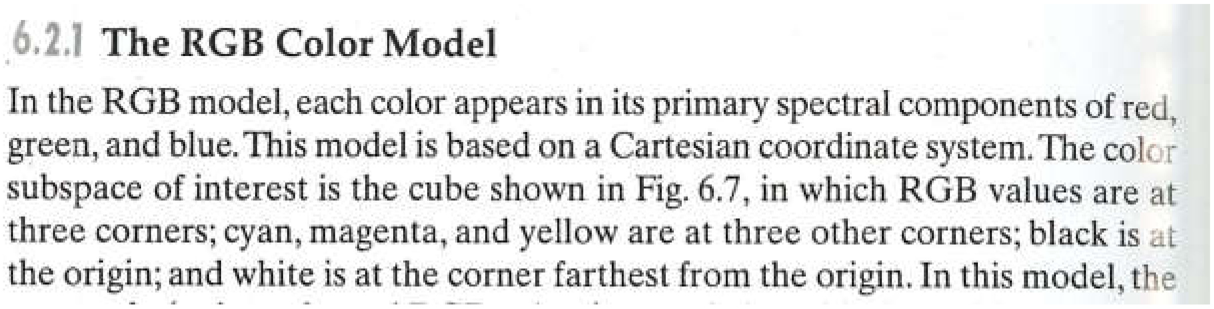
The diagram is useful for color mixing, e.g.

- a straight line joining any two points defines all colors generated by adding those colors,
- in particular, if one of these points is reference white and the other is some color on the boundary, then the colors on the line in-between represent all the shades of that particular spectrum color

Color model

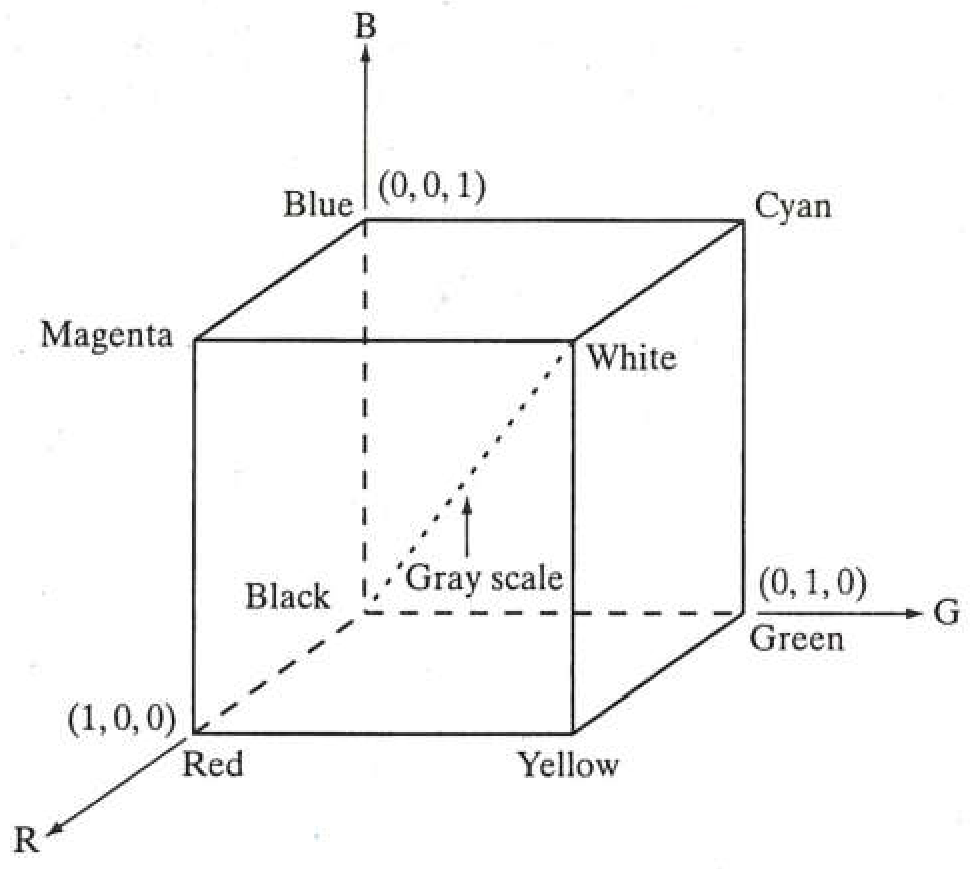
Two important aspects to retain about color models:

1. conversion between color models can be either linear or nonlinear,
2. some models can be more useful as they can decouple color and gray-scale components of a color image, e.g. HSI, YUV.



6.2.1 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 values are at three corners; 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



Pixel depth refers to the number of bits used to represent each pixel in the RGB space

If each pixel component (red, green and blue) is represented by 8 bits, the pixel is said to have a depth of 24 bits.

A **full-color** image refers to a 24-bit RGB color image. The number of possible colors in a full-color image is:

$$(2^8)^3 = 16{,}777{,}216 \text{ colors (or 16 million colors)} \qquad 6.27$$

6.2.2 The CMY and CMYK Color Models

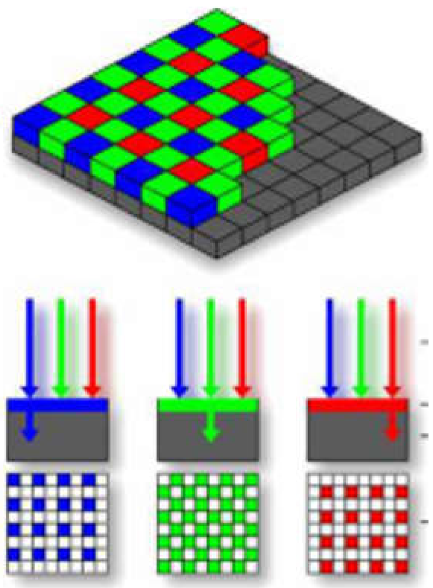
As indicated in Section 6.1, cyan, magenta, and yellow are the secondary colors of light or, alternatively, the primary colors of pigments. For example, when a surface coated with cyan pigment is illuminated with white light, no red light is reflected from the surface. That is, cyan subtracts red light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

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

Similarly, pure magenta does not reflect green, and pure yellow does not reflect blue. Equation (6.2-1) also reveals that RGB values can be obtained easily from

RGB Image Formation in Cameras

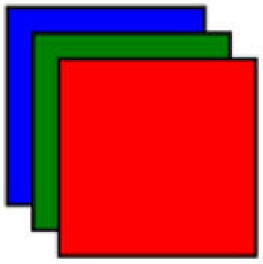
The Bayer Filter



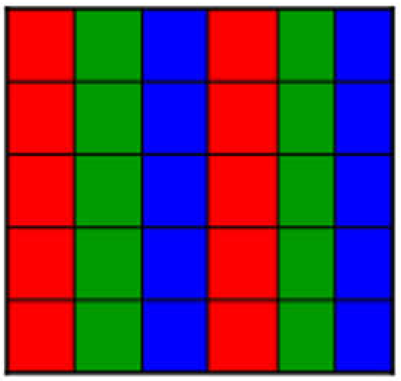
- Based upon the observation that human vision is much more responsive to green light than red or blue
- Half the pixels in the CCD are allocated to green, ¼ to red and ¼ to blue
- Color is generated for the whole CCD by interpolating neighbor values

RGB Image Format

- Images pixels can be either *planar* or *packed* format
- Planar format separates the colors into three contiguous arrays in memory
- Packed alternate R->G->B->R->... in memory



Planar



Packed

HSI Color Model

Although RGB and CMY color models are very well suited for hardware and RGB reflects well the sensitivity of the human eye to these primary colors, both are not suited for describing color in a way that is easily interpreted by humans.

When humans see a color object, they tend to describe it by its **hue**, **saturation** and **brightness**, i.e. HSI model is used

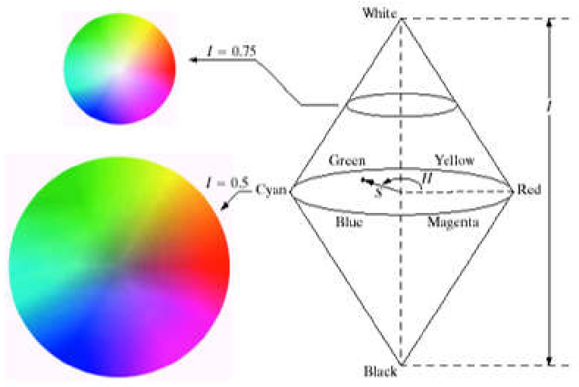
In addition, HSI decouples brightness from the chroma components.

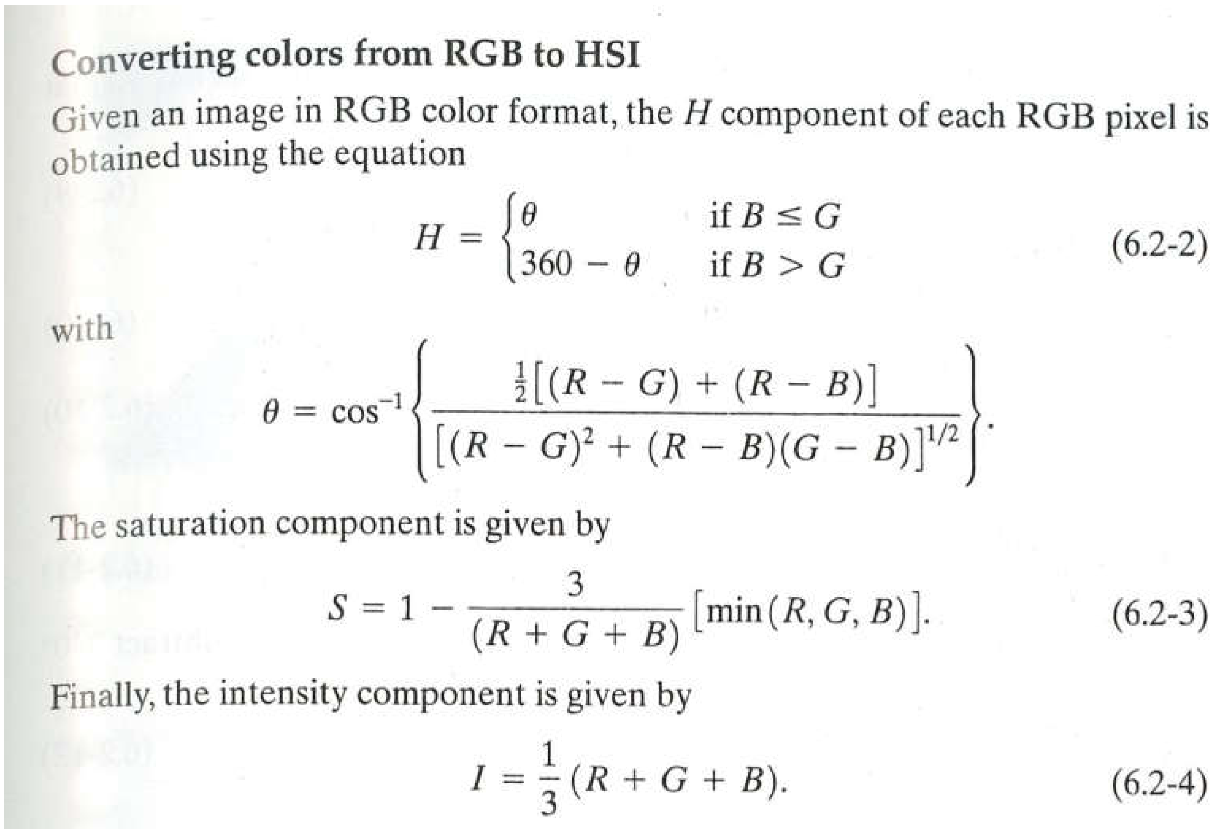
Three Perceptual Measures

1. ***Intensity*** *or* ***Value*** *or* **Brightness**: varies along the vertical axis and measures the extent to which an area appears to exhibit light. It is proportional to the electromagnetic energy radiated by the source.

2. ***Hue****:* denoted by H and varies along the circumference. It measures the extent to which an area matches colors red, orange, yellow, blue or purple (or a mixture of any two). In other words, hue is a parameter which distinguishes the color of the source, i.e., is the color red, yellow, blue, etc.

3. **Saturation**: the quantity which distinguishes a pure spectral light from a pastel shade of the same hue. It is simply a measure of white light added to the pure spectral color. In other words, saturation is the colorfulness of an area judged in proportion to the brightness of the object itself. Saturation varies along the radial axis.





Converting colors from RGB to HSI

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

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases} \qquad (6.2\text{-}2)$$

with

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

The saturation component is given by

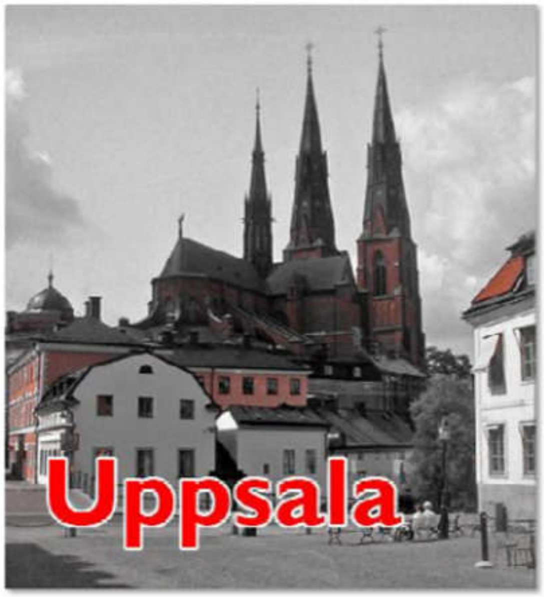
$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]. \qquad (6.2\text{-}3)$$

Finally, the intensity component is given by

$$I = \frac{1}{3}(R+G+B). \qquad (6.2\text{-}4)$$

Examples:

Masking out a region with similiar color



•Find the pixels in the range of the desired color in the **Hue**-channel

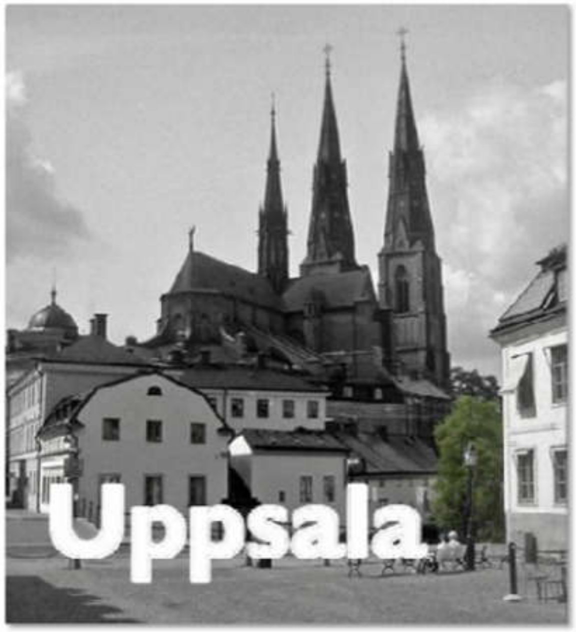
•Set all other pixels to **o** in the **Saturation**-channel

```
%imgHSV : HSV-transformed image
%masked H-value: 0-0.1 and 0.9-1.0
mask = (imgHSV(:,:,1) < 0.1)
         + (imgHSV(:,:,1)> 0.9);
S  = imgHSV(:,:,2) .* mask;

imgHSV_r (:,:,3) = imgHSV(:,:,3);
imgHSV_r (:,:,2) = S;
imgHSV_r (:,:,1) = imgHSV(:,:,1);
```

Keep the reddish colors 6.6

Another example



- Find the pixels in the range of the desired color in the **Hue**-channel
- Set all other pixels to **0** in the **Saturation**-channel

```
%imgHSV : HSV-transformed image
%H-value of the tree : 0.15-0.27
mask = (imgHSV(:,:,1) < 0.15)
          + (imgHSV(:,:,1)> 0.27);
S   = imgHSV(:,:,2) .* (1-mask);

imgHSV_g (:,:,3) = imgHSV(:,:,3);
imgHSV_g (:,:,2) = S;
imgHSV_g (:,:,1) = imgHSV(:,:,1);
```

Keep the greenish colors

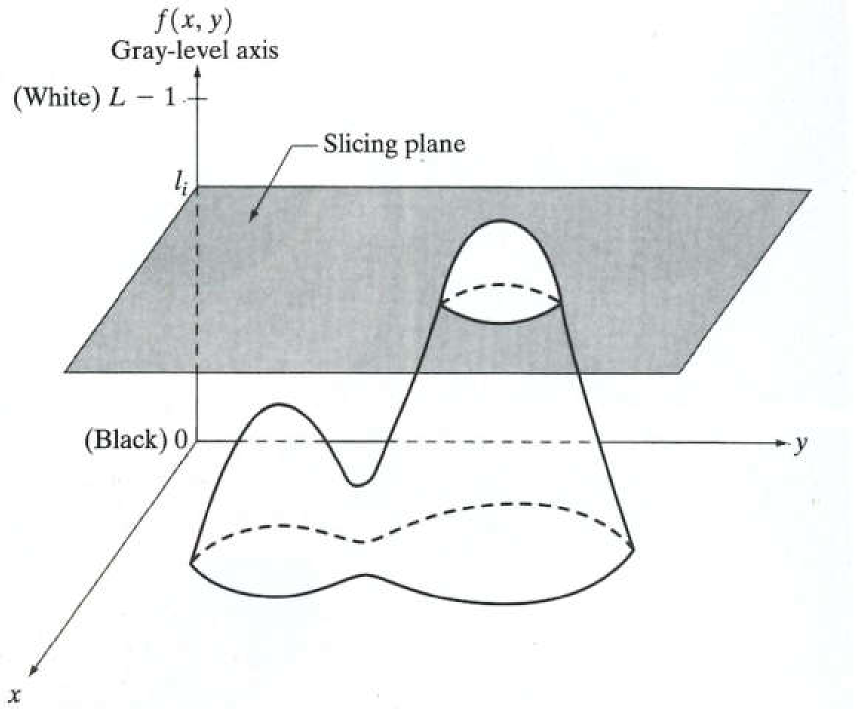
6.3 Pseudocolor Image Processing

Pseudocolor (also called *false color*) image processing consists of assigning colors to gray values based on a specified criterion. The term *pseudo* or *false* color

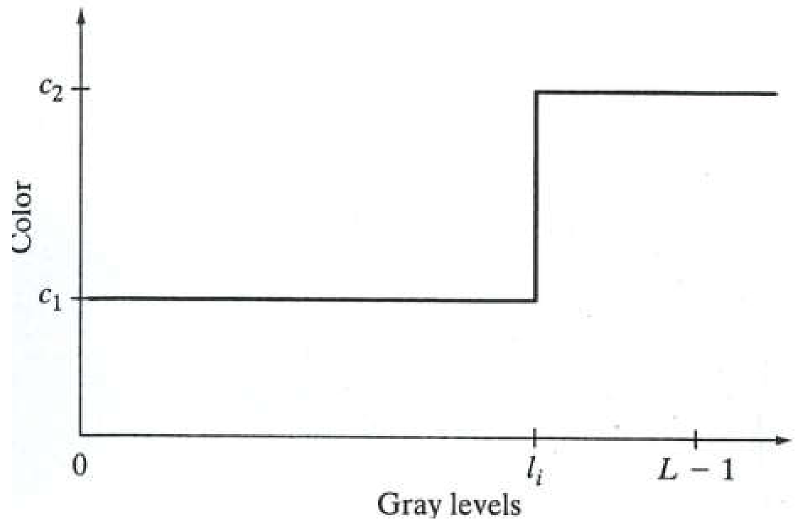
6.3.1 Intensity Slicing

The technique of *intensity* (sometimes called *density*) *slicing* and color coding is one of the simplest examples of pseudocolor image processing. If an image is interpreted as a 3-D function (intensity versus spatial coordinates), 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

If a different color is assigned to each side of the plane shown in Fig. 6.18, any pixel whose gray 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



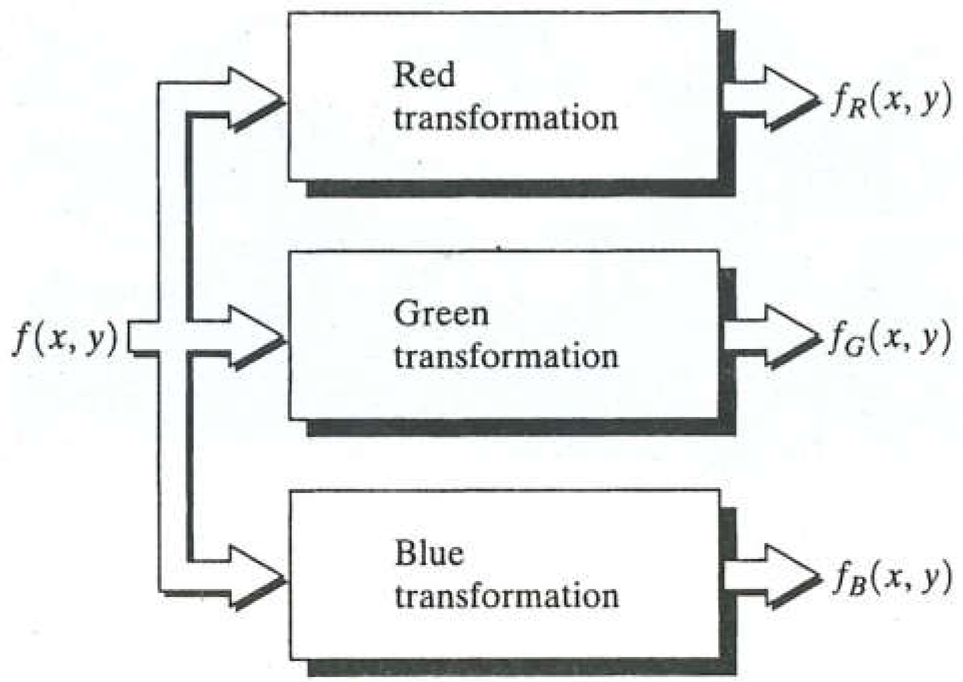
defines the same mapping as in Fig. 6.18. According to the mapping function shown in Fig. 6.19, any input gray level is assigned one of two colors, depending on whether it is above or below the value of l_i. When more levels are used,



6.3.2 Gray Level to Color Transformations

attractive is shown in Fig. 6.23. Basically, the idea underlying this approach is to perform three independent transformations on the gray level of any input pixel. The three results are then fed separately into the red, green, and blue channels of a color television monitor. This method produces a composite image whose color content is modulated by the nature of the transformation functions. Note

just described. There, piecewise linear functions of the gray levels (Fig. 6.19) are used to generate colors. The method discussed in this section, on the other



Color Image Processing: multi-spectral images

Many images are multispectral, i.e. they have been acquired by different sensors at different wavelengths. Combining them to obtain a color image can be achieved as follows:

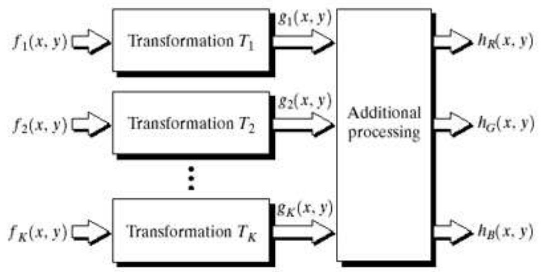


FIGURE 6.26 A pseudocolor coding approach used when several monochrome images are available.

additional processing may include color balancing, combining images, and selecting three of them for display, etc.

6.4 Basics of Full-Color Image Processing

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 component image individually and then form a composite processed 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 compo-

6.5 Color Transformations

1.foundation:

ment it. Suppose, for example, that we wish to modify the intensity of the image in Fig. 6.30(a) using

$$g(x, y) = kf(x, y) \tag{6.5-3}$$

where $0 < k < 1$. In the HSI color space, this can be done with the simple transformation

$$s_3 = kr_3 \tag{6.5-4}$$

where $s_1 = r_1$ and $s_2 = r_2$. Only HSI intensity component r_3 is modified. In the RGB color space, three components must be transformed:

$$s_i = kr_i \qquad i = 1, 2, 3. \tag{6.5-5}$$

The CMY space requires a similar set of linear transformations:

$$s_i = kr_i + (1 - k) \qquad i = 1, 2, 3. \tag{6.5-6}$$

Color Image Processing: color slicing

Idea: highlight a range of colors in an image in order to

- separate them from background, or
- use the region defined by color mask for further processing, e.g. segmentation

This is a complex extension of gray level slicing due to the multi-valued nature of color images

How can this be done? The simplest way to slice a color image is to map the colors outside some range of interest to some neutral color and leave the rest as they are. Let $w=(a_1,a_2,a_3)$ be the average of the color region of interest and W the width of this region, then

$$s_i = \begin{cases} 0{,}5 & if \left[|r_j - a_j| > \frac{W}{2} \right]_{\forall 1 \le j \le 3} \\ r_i \end{cases} \qquad \text{for } i=1,2,3$$

If a sphere is used to specify the region of interest, then

$$s_i = \begin{cases} 0{,}5 & if \sum_{j=1}^{3} (r_j - a_j)^2 \succ R_0^2 \\ r_i \end{cases}$$

Color Image Processing: Tone and color corrections

Goal: correct color image through pixel transformations to get a better visualization and / or print out.

L*a*b* color space is perceptually uniform, i.e. color differences are perceived uniformly.

Like HSI, L*a*b* decouples intensity from color

Example: tonal correction for three common tonal imbalances: flat, light and dark images, see images next.

Figure 6.35 shows typical 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 [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.

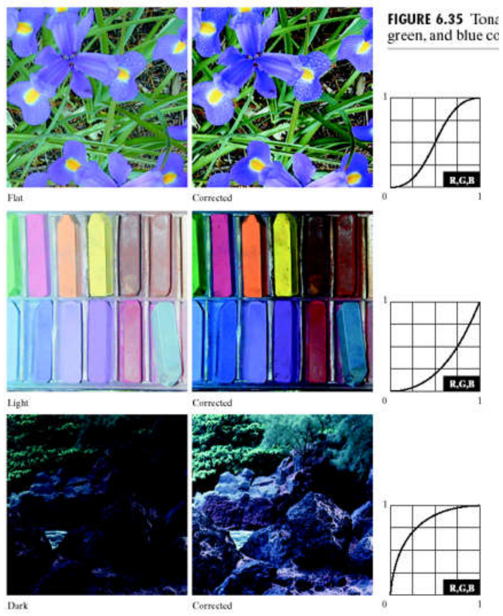
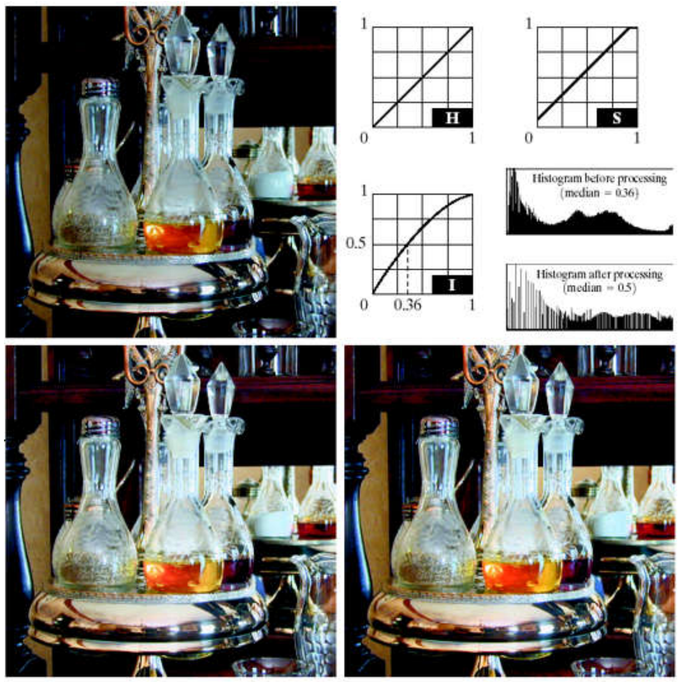


FIGURE 6.35 Tona
green, and blue co

6.5.5 Histogram Processing

component and/or histogram. As might be expected, it is generally unwise to histogram equalize the components 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 ex-



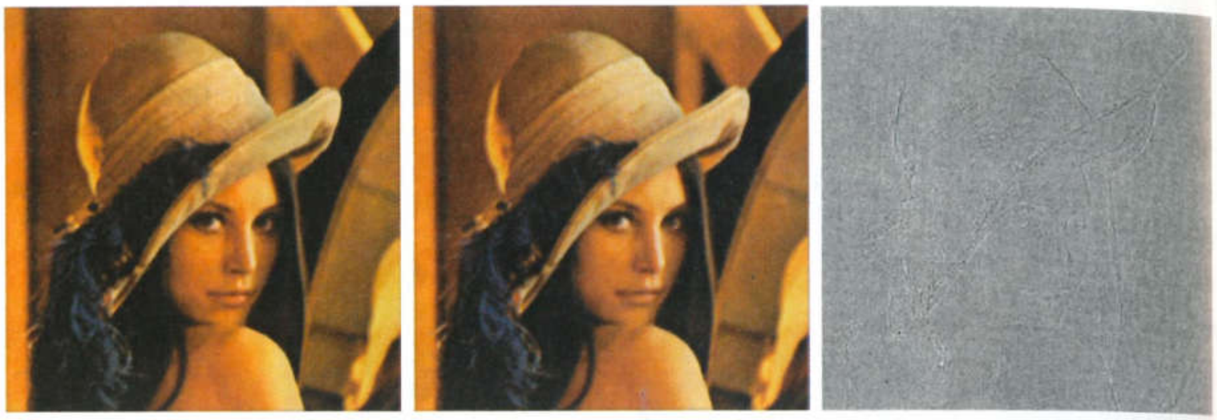
H
S
I
0
1
0.5
0.36
Histogram before processing (median = 0.36)
Histogram after processing (median = 0.5)

Answer: This is a color histogram equalization example.

a) is the original color image (1 point)
b) histogram equalization operation in HSI: H is kept unchanged in order to preserve colors, I is enhanced through a grey-level transformation (e.g. n'th root) in order to make darker objects lighter; and S is slightly adjusted to compensate for changes in the intensity component. (3 points)
c) result of equalizing the I component only; colors have changed slightly due to changes in the intensity. (2 points)
d) result of adjusting the saturation component in order to compensate for the changes in intensity. (2 points)

6.6 Smoothing and Sharpening

6.6.1 Color Image Smoothing



a b c

using the 5×5 gray-level averaging mask of Section 3.6. We simply smooth independently each of the RGB color planes and then combine the processed planes to form a smoothed full-color result. The image so computed is shown

In Section 6.2 it was noted that an important advantage of the HSI color model is that it decouples intensity (closely related to gray scale) and color information. This makes it suitable for many gray-scale processing techniques and suggests that it might be more efficient to smooth only the intensity component of the HSI representation in Fig. 6.39. 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

the difference image in Fig. 640(c), is not identical. This is due to the fact that the average of two pixels of differing color is a mixture of the two colors, not either of the original colors. By smoothing only the intensity image, the pixels in Fig. 6.40(b) maintain their original hue and saturation—and thus their original color.

6.6.2 Color Image Sharpening

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.

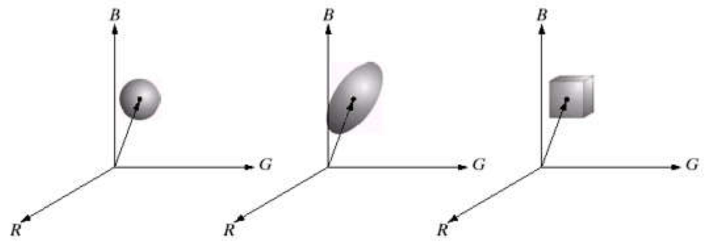
▣ Figure 6.41(a) was obtained using Eq. (3.7-6) to compute the Laplacians of the RGB component images in Fig. 6.38 and combining them to produce the sharpened full-color result. Figure 6.41(b) shows a similarly sharpened image based on the HSI components in Fig. 6.39. This result was generated by combining the Laplacian of the intensity component with the unchanged hue and saturation

6.7 Color Segmentation

6.7.2 Segmentation in RGB Vector Space

Color Image Processing: Segmentation

Suppose that regions of specific color range are to be segmented. The specific color is specified by an average color **a** and a neighborhood around it, defined by a suitable distance measure: we say that z is similar to a if D(a,z) is smaller than a threshold D_0.



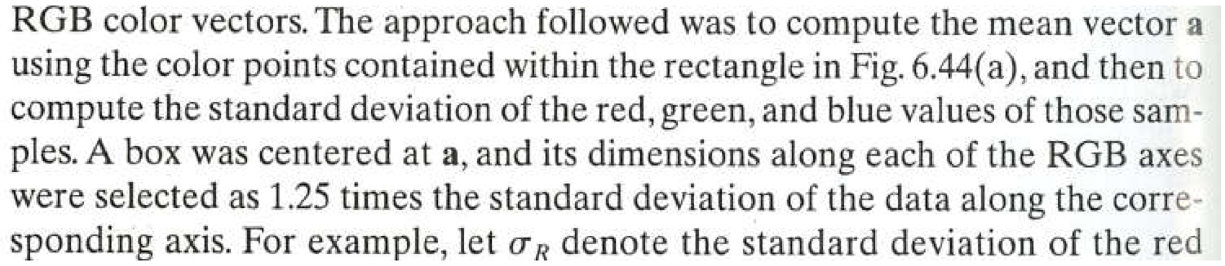
a b c

FIGURE 6.43 Three approaches for enclosing data regions for RGB vector segmentation.

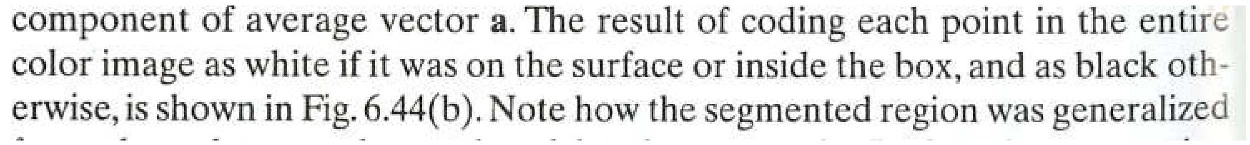
Bounding box approach:

ing box, as illustrated in Fig. 6.43(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. Computation of the standard deviations is done only once using sample color data.

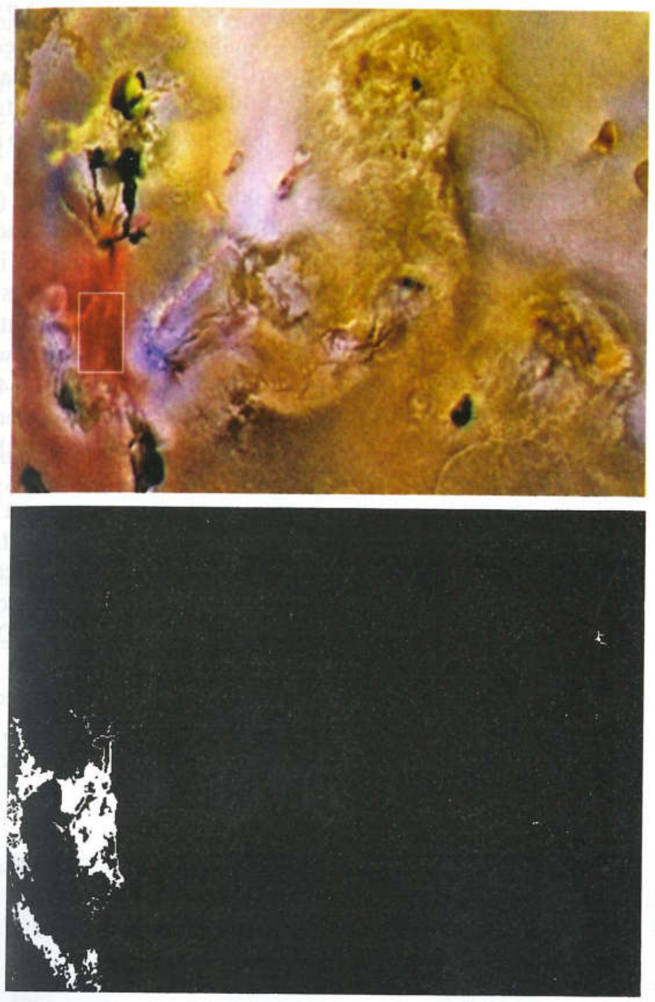
We Want to segment reddish colors in the image, as in the rectangular region.



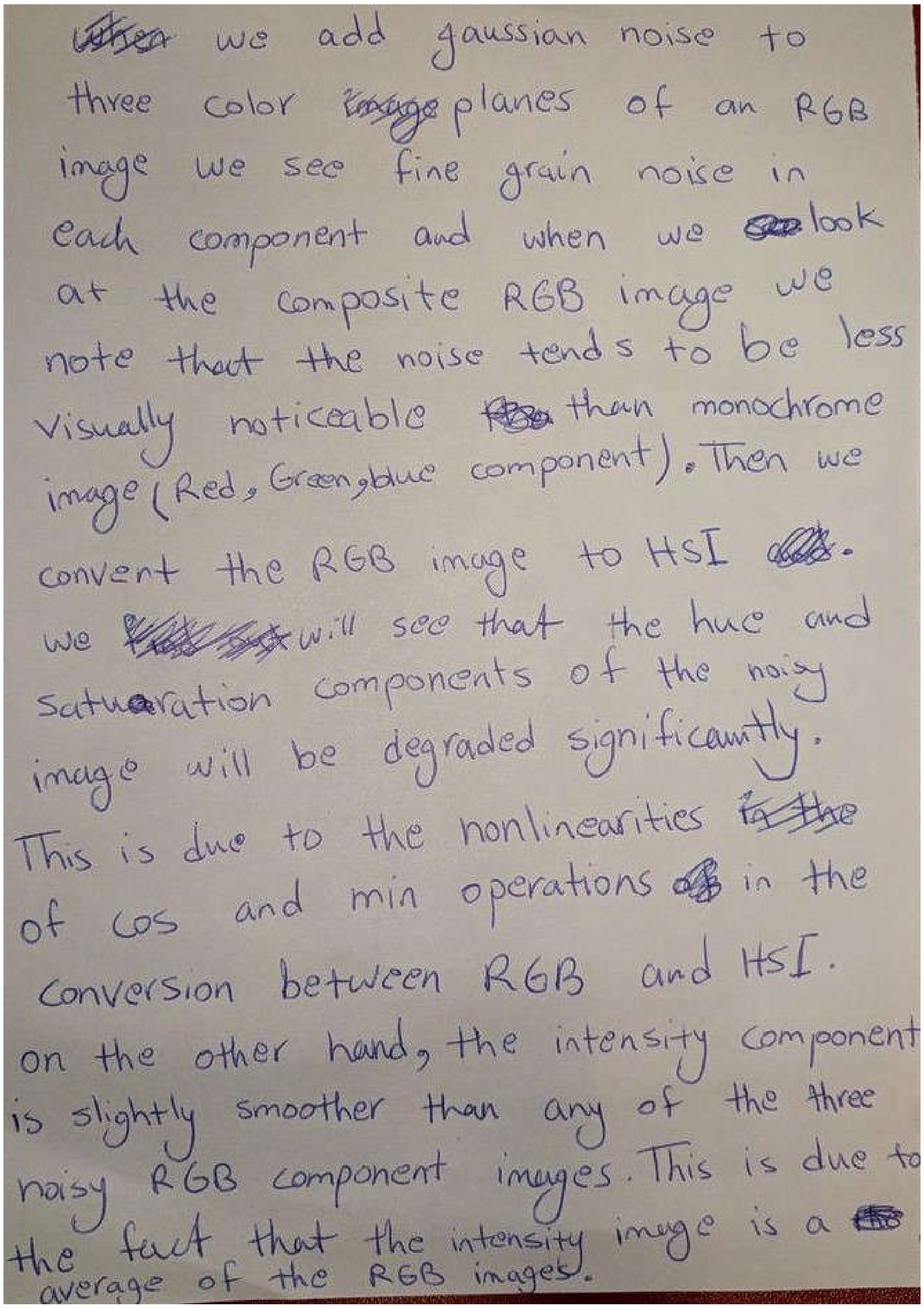
RGB color vectors. The approach followed was to compute the mean vector **a** using the color points contained within the rectangle in Fig. 6.44(a), and then to compute the standard deviation of the red, green, and blue values of those samples. A box was centered at **a**, and its dimensions along each 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



component of average vector **a**. The result of coding each point in the entire color image as white if it was on the surface or inside the box, and as black otherwise, is shown in Fig. 6.44(b). Note how the segmented region was generalized



6.8 Noise in Color Images



~~When~~ we add gaussian noise to three color ~~image~~ planes of an RGB image we see fine grain noise in each component and when we ~~see~~ look at the composite RGB image we note that the noise tends to be less visually noticeable ~~RGB~~ than monochrome image (Red, Green, blue component). Then we convert the RGB image to HSI [illegible]. we [illegible] will see that the hue and saturation components of the noisy image will be degraded significantly. This is due to the nonlinearities ~~in the~~ of cos and min operations [illegible] in the conversion between RGB and HSI. on the other hand, the intensity component is slightly smoother than any of the three noisy RGB component images. This is due to the fact that the intensity image is a ~~the~~ average of the RGB images.