14 Colorimetric Color Spaces

In any application that requires precise, reproducible, and device-independent presentation of colors, the use of calibrated color systems is an absolute necessity. For example, color calibration is routinely used throughout the digital print work flow but also in digital film production, professional photography, image databases, etc. One may have experienced how difficult it is, for example, to render a good photograph on a color laser printer, and even the color reproduction on monitors largely depends on the particular manufacturer and computer system.

All the color spaces described in Chapter 12, Sec. 12.2, somehow relate to the physical properties of some media device, such as the specific colors of the phosphor coatings inside a CRT tube or the colors of the inks used for printing. To make colors appear similar or even identical on different media modalities, we need a representation that is independent of how a particular device reproduces these colors. Color systems that describe colors in a measurable, device-independent fashion are called *colorimetric* or *calibrated*, and the field of *color science* is traditionally concerned with the properties and application of these color systems (see, e.g., [258] or [215] for an overview). While several colorimetric standards exist, we focus on the most widely used CIE systems in the remaining part of this section.

14.1 CIE Color Spaces

The XYZ color system, developed by the CIE (Commission Internationale d'Èclairage)[1] in the 1920s and standardized in 1931, is the foundation of most colorimetric color systems that are in use today [195, p. 22].

[1] International Commission on Illumination (www.cie.co.at).

14.1.1 CIE XYZ Color Space

The CIE XYZ color scheme was developed after extensive measurements of human visual perception under controlled conditions. It is based on three imaginary primary colors X, Y, Z, which are chosen such that all visible colors can be described as a summation of positive-only components, where the Y component corresponds to the perceived lightness or *luminosity* of a color. All visible colors lie inside a 3D cone-shaped region (Fig. 14.1(a)), which interestingly enough does not include the primary colors themselves.

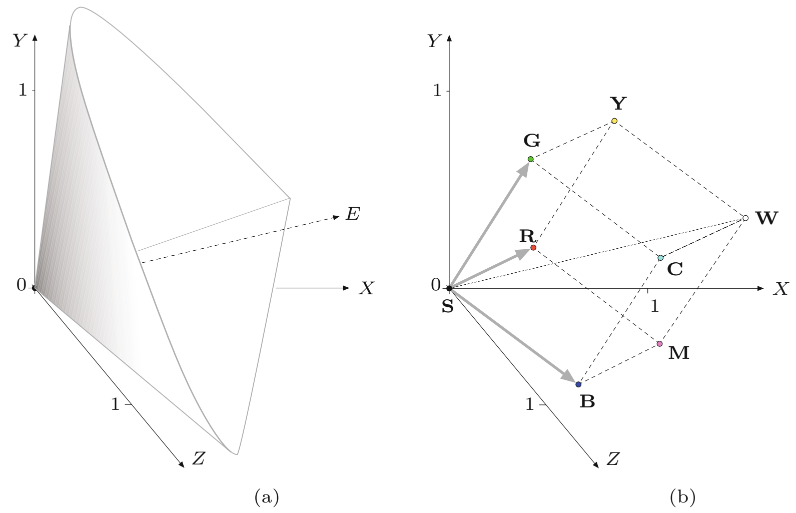


Fig. 14.1 The XYZ color space is defined by the three imaginary primary colors X, Y, Z, where the Y dimension corresponds to the perceived luminance. All visible colors are contained inside an open, cone-shaped volume that originates at the black point **S** (a), where E denotes the axis of neutral (gray) colors. The RGB color space maps to the XYZ space as a linearly distorted cube (b). See also Fig. 14.5(a).

Some common color spaces, and the RGB color space in particular, conveniently relate to XYZ space by a *linear* coordinate transformation, as described in Sec. 14.4. Thus, as shown in Fig. 14.1(b), the RGB color space is embedded in the XYZ space as a distorted cube, and therefore straight lines in RGB space map to straight lines in XYZ again. The CIE XYZ scheme is (similar to the RGB color space) *nonlinear* with respect to human visual perception, that is, a particular fixed distance in XYZ is not perceived as a uniform color change throughout the entire color space. The XYZ coordinates of the RGB color cube (based on the primary colors defined by ITU-R BT.709) are listed in Table 14.1.

14.1.2 CIE x, y Chromaticity

As mentioned, the luminance in XYZ color space increases along the Y axis, starting at the black point **S** located at the coordinate origin ($X = Y = Z = 0$). The color hue is independent of the luminance and thus independent of the Y value. To describe the corresponding "pure" color hues and saturation in a convenient manner, the CIE system also defines the three *chromaticity* values

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}, \tag{14.1}$$

where (obviously) $x+y+z = 1$ and thus one of the three values (e.g., z) is redundant. Equation (14.1) describes a central projection from

Pt.	Color	R	G	B	X	Y	Z	x	y
S	Black	0.00	0.00	0.00	0.0000	0.0000	0.0000	0.3127	0.3290
R	Red	1.00	0.00	0.00	0.4125	0.2127	0.0193	0.6400	0.3300
Y	Yellow	1.00	1.00	0.00	0.7700	0.9278	0.1385	0.4193	0.5052
G	Green	0.00	1.00	0.00	0.3576	0.7152	0.1192	0.3000	0.6000
C	Cyan	0.00	1.00	1.00	0.5380	0.7873	1.0694	0.2247	0.3288
B	Blue	0.00	0.00	1.00	0.1804	0.0722	0.9502	0.1500	0.0600
M	Magenta	1.00	0.00	1.00	0.5929	0.2848	0.9696	0.3209	0.1542
W	White	1.00	1.00	1.00	0.9505	1.0000	1.0888	0.3127	0.3290

Table 14.1
Coordinates of the RGB color cube in CIE XYZ space. The X, Y, Z values refer to standard (ITU-R BT. 709) primaries and white point D65 (see Table 14.2), x, y denote the corresponding CIE chromaticity coordinates.

X, Y, Z coordinates onto the 3D plane

$$X + Y + Z = 1, \tag{14.2}$$

with the origin **S** as the projection center (Fig. 14.2). Thus, for an arbitrary XYZ color point $\mathbf{A} = (X_a, Y_a, Z_a)$, the corresponding chromaticity coordinates $\mathbf{a} = (x_a, y_a, z_a)$ are found by intersecting the line $\overline{\mathbf{SA}}$ with the $X + Y + Z = 1$ plane (Fig. 14.2(a)). The final x, y coordinates are the result of projecting these intersection points onto the X/Y-plane (Fig. 14.2(b)) by simply dropping the Z component z_a.

The result is the well-known horseshoe-shaped *CIE x, y chromaticity diagram*, which is shown in Fig. 14.2(c). Any x, y point in this diagram defines the hue and saturation of a particular color, but only the colors inside the horseshoe curve are potentially visible. Obviously an infinite number of X, Y, Z colors (with different luminance values) project to the same x, y, z chromaticity values, and the XYZ color coordinates thus cannot be uniquely reconstructed from given chromaticity values. Additional information is required. For example, it is common to specify the visible colors of the CIE system in the form Yxy, where Y is the original luminance component of the XYZ color. Given a pair of chromaticity values x, y (with $y > 0$) and an arbitrary Y value, the missing X, Z coordinates are obtained (using the definitions in Eqn. (14.1)) as

$$X = x \cdot \frac{Y}{y}\,, \qquad Z = z \cdot \frac{Y}{y} = (1 - x - y) \cdot \frac{Y}{y}. \tag{14.3}$$

The CIE diagram not only yields an intuitive layout of color hues but exhibits some remarkable formal properties. The xy values along the outer horseshoe boundary correspond to monochromatic ("spectrally pure"), maximally saturated colors with wavelengths ranging from below 400 nm (purple) up to 780 nm (red). Thus the position of any color inside the xy diagram can be specified with respect to any of the primary colors at the boundary, except for the points on the connecting line ("purple line") between 380 and 780 nm, whose purple hues do not correspond to primary colors but can only be generated by mixing other colors.

The *saturation* of colors falls off continuously toward the "neutral point" (**E**) at the center of the horseshoe, with $x = y = \frac{1}{3}$ (or $X = Y = Z = 1$, respectively) and zero saturation. All other colorless (i.e., gray) values also map to the neutral point, just as any set of colors

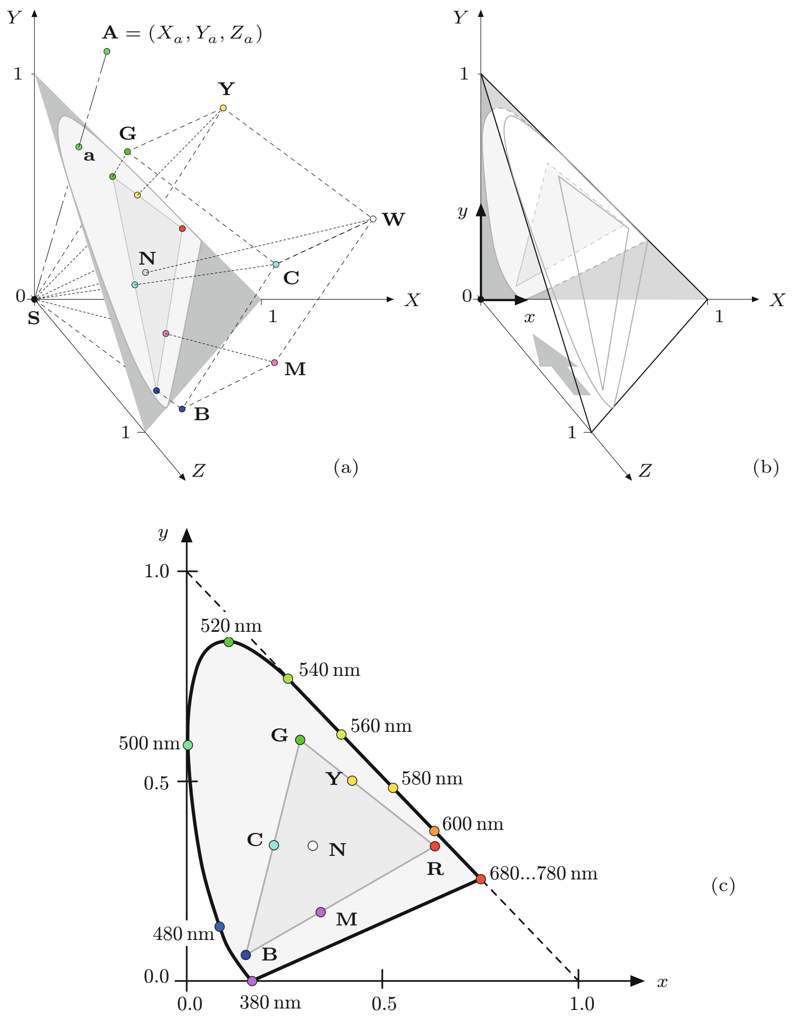


Fig. 14.2
CIE x, y chromaticity diagram. For an arbitrary XYZ color point $\mathbf{A} = (X_a, Y_a, Z_a)$, the chromaticity values $\mathbf{a} = (x_a, y_a, z_a)$ are obtained by a central projection onto the 3D plane $X + Y + Z = 1$ (a). The corner points of the RGB cube map to a triangle, and its white point $\mathbf{W}$ maps to the (colorless) neutral point $\mathbf{E}$. The intersection points are then projected onto the X/Y plane (b) by simply dropping the Z component, which produces the familiar CIE chromaticity diagram shown in (c). The CIE diagram contains all visible color tones (hues and saturations) but no luminance information, with wavelengths in the range 380–780 nanometers. A particular color space is specified by at least three primary colors (tristimulus values; e.g., $\mathbf{R}$, $\mathbf{G}$, $\mathbf{B}$), which define a triangle (linear hull) containing all representable colors.

with the same hue but different brightness corresponds to a single x, y point. All possible composite colors lie inside the convex hull specified by the coordinates of the primary colors of the CIE diagram and, in particular, complementary colors are located on straight lines that run diagonally through the white point.

14.1.3 Standard Illuminants

A central goal of colorimetry is the quantitative measurement of colors in physical reality, which strongly depends on the color properties of the illumination. The CIE system specifies a number of standard illuminants for a variety of real and hypothetical light sources, each specified by a spectral radiant power distribution and the "correlated color temperature" (expressed in degrees Kelvin) [258, Sec. 3.3.3]. The following daylight (D) illuminants are particularly important for the design of digital color spaces (Table 14.2):

D50 emulates the spectrum of natural (direct) sunlight with an equivalent color temperature of approximately 5000° K. D50 is the recommended illuminant for viewing reflective images, such as paper prints. In practice, D50 lighting is commonly implemented with fluorescent lamps using multiple phosphors to approximate the specified color spectrum.

D65 has a correlated color temperature of approximately 6500° K and is designed to emulate the average (indirect) daylight observed under an overcast sky on the northern hemisphere. D65 is also used as the reference white for emittive devices, such as display screens.

The standard illuminants serve to specify the ambient viewing light but also to define the reference white points in various color spaces in the CIE color system. For example, the sRGB standard (see Sec. 14.4) refers to D65 as the media white point and D50 as the ambient viewing illuminant. In addition, the CIE system also specifies the range of admissible viewing angles (commonly at ±2°).

	°K	X	Y	Z	x	y
D50	5000	0.96429	1.00000	0.82510	0.3457	0.3585
D65	6500	0.95045	1.00000	1.08905	0.3127	0.3290
N	—	1.00000	1.00000	1.00000	$0.333\dot{3}$	$0.333\dot{3}$

Table 14.2
CIE color parameters for the standard illuminants **D50** and **D65**. **E** denotes the absolute neutral point in CIE XYZ space.

14.1.4 Gamut

The set of all colors that can be handled by a certain media device or can be represented by a particular color space is called "gamut". This is usually a contiguous region in the 3D CIE XYZ color space or, reduced to the representable color hues and ignoring the luminance component, a convex region in the 2D CIE chromaticity diagram.

Figure 14.3 illustrates some typical gamut regions inside the CIE diagram. The gamut of an output device mainly depends on the technology employed. For example, ordinary color monitors are typically not capable of displaying all colors of the gamut covered by the corresponding color space (usually sRGB). Conversely, it is also possible that devices would reproduce certain colors that cannot be represented in the utilized color space. Significant deviations exist, for example, between the RGB color space and the gamuts associated with CMYK-based printers. Also, media devices with very large gamuts exist, as demonstrated by the laser display system in Fig. 14.3. Representing such large gamuts and, in particular, transforming between different color representations requires adequately sized color spaces, such as the Adobe-RGB color space or CIELAB (described in Sec. 14.2), which covers the entire visible portion of the CIE diagram.

14.1.5 Variants of the CIE Color Space

The original CIEXYZ color space and the derived xy chromaticity diagram have the disadvantage that color differences are not perceived equally in different regions of the color space. For example,

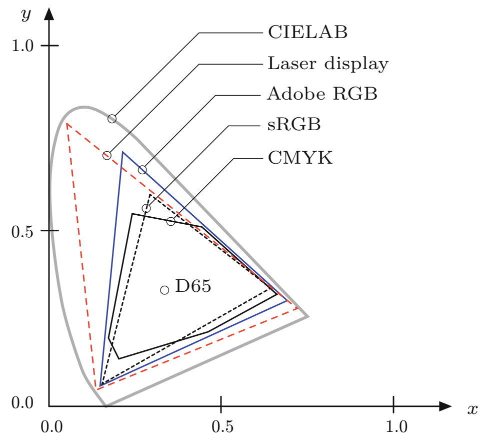


Fig. 14.3
Gamut regions for different color spaces and output devices inside the CIE diagram.

large color changes are perceived in the *magenta* region for a given shift in XYZ while the change is relatively small in the *green* region for the same coordinate distance. Several variants of the CIE color space have been developed for different purposes, primarily with the goal of creating perceptually uniform color representations without sacrificing the formal qualities of the CIE reference system. Popular CIE-derived color spaces include CIE YUV, YU′V′, $\mathrm{YC_bC_r}$, and particularly CIELAB and CIELUV, which are described in the following sections. In addition, CIE-compliant specifications exist for most common color spaces (see Ch. 12, Sec. 12.2), which allow more or less dependable conversions between almost any pair of color spaces.

14.2 CIELAB

The CIELAB color model (specified by CIE in 1976) was developed with the goal of linearizing the representation with respect to human color perception and at the same time creating a more intuitive color system. Since then, CIELAB[2] has become a popular and widely used color model, particularly for high-quality photographic applications. It is used, for example, inside Adobe Photoshop as the standard model for converting between different color spaces. The dimensions in this color space are the luminosity L^* and the two color components a^*, b^*, which specify the color hue and saturation along the *green-red* and *blue-yellow* axes, respectively. All three components are *relative* values and refer to the specified reference white point $\mathbf{C}_{\mathrm{ref}} = (X_{\mathrm{ref}}, Y_{\mathrm{ref}}, Z_{\mathrm{ref}})$. In addition, a nonlinear correction function (similar to the modified gamma correction described in Ch. 4, Sec. 4.7.6) is applied to all three components, as will be detailed further.

14.2.1 CIEXYZ→CIELAB Conversion

Several specifications for converting to and from CIELAB space exist that, however, differ marginally and for very small L values only. The

[2] Often CIELAB is simply referred to as the “Lab” color space.

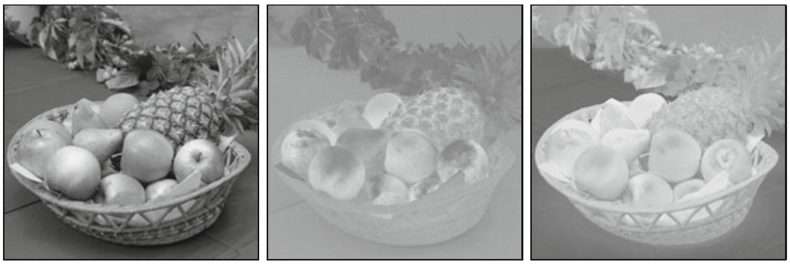


Fig. 14.4
CIELAB components shown as grayscale images. The contrast of the a^* and b^* images has been increased by 40% for better viewing.

current specification for converting between CIEXYZ and CIELAB colors is defined by ISO Standard 13655 [120] as follows:

$$L^* = 116 \cdot Y' - 16, \tag{14.4}$$
$$a^* = 500 \cdot (X' - Y'), \tag{14.5}$$
$$b^* = 200 \cdot (Y' - Z'), \tag{14.6}$$

with

$$X' = f_1\big(\tfrac{X}{X_{\mathrm{ref}}}\big), \quad Y' = f_1\big(\tfrac{Y}{Y_{\mathrm{ref}}}\big), \quad Z' = f_1\big(\tfrac{Z}{Z_{\mathrm{ref}}}\big), \tag{14.7}$$
$$f_1(c) = \begin{cases} c^{1/3} & \text{for } c > \epsilon, \\ \kappa \cdot c + \frac{16}{116} & \text{for } c \le \epsilon, \end{cases} \tag{14.8}$$

and

$$\epsilon = \big(\tfrac{6}{29}\big)^3 = \tfrac{216}{24389} \approx 0.008856, \tag{14.9}$$
$$\kappa = \tfrac{1}{116}\big(\tfrac{29}{3}\big)^3 = \tfrac{841}{108} \approx 7.787. \tag{14.10}$$

For the conversion in Eqn. (14.7), D65 is usually specified as the reference white point $\mathbf{C}_{\mathrm{ref}} = (X_{\mathrm{ref}}, Y_{\mathrm{ref}}, Z_{\mathrm{ref}})$, that is, $X_{\mathrm{ref}} = 0.95047$, $Y_{\mathrm{ref}} = 1.0$ and $Z_{\mathrm{ref}} = 1.08883$ (see Table 14.2). The L^* values are positive and typically in the range $[0, 100]$ (often scaled to $[0, 255]$), but may theoretically be greater. Values for a^* and b^* are in the range $[-127, +127]$. Figure 14.4 shows the separation of a color image into the corresponding CIELAB components. Table 14.3 lists the relation between CIELAB and XYZ coordinates for selected RGB colors. The given $R'G'B'$ values are (nonlinear) sRGB coordinates with D65 as the reference white point.[3] Figure 14.5(c) shows the transformation of the RGB color cube into the CIELAB color space.

14.2.2 CIELAB→CIEXYZ Conversion

The reverse transformation from CIELAB space to CIEXYZ coordinates is defined as follows:

$$X = X_{\mathrm{ref}} \cdot f_2\big(L' + \tfrac{a^*}{500}\big), \tag{14.11}$$
$$Y = Y_{\mathrm{ref}} \cdot f_2\big(L'\big), \tag{14.12}$$
$$Z = Z_{\mathrm{ref}} \cdot f_2\big(L' - \tfrac{b^*}{200}\big), \tag{14.13}$$

[3] Note that sRGB colors in Java are specified with respect to white point D50, which explains certain numerical deviations (see Sec. 14.7).

Table 14.3
CIELAB coordinates for selected color points in sRGB. The sRGB components R', G', B' are nonlinear (i.e., gamma-corrected), white point is D65 (see Table 14.2).

		sRGB			CIEXYZ (D65)			CIELAB		
Pt.	Color	R'	G'	B'	X_{65}	Y_{65}	Z_{65}	L^*	a^*	b^*
S	Black	0.00	0.00	0.00	0.0000	0.0000	0.0000	0.00	0.00	0.00
R	Red	1.00	0.00	0.00	0.4125	0.2127	0.0193	53.24	80.09	67.20
Y	Yellow	1.00	1.00	0.00	0.7700	0.9278	0.1385	97.14	−21.55	94.48
G	Green	0.00	1.00	0.00	0.3576	0.7152	0.1192	87.74	−86.18	83.18
C	Cyan	0.00	1.00	1.00	0.5380	0.7873	1.0694	91.11	−48.09	−14.13
B	Blue	0.00	0.00	1.00	0.1804	0.0722	0.9502	32.30	79.19	−107.86
M	Magenta	1.00	0.00	1.00	0.5929	0.2848	0.9696	60.32	98.24	−60.83
W	White	1.00	1.00	1.00	0.9505	1.0000	1.0888	100.00	0.00	0.00
K	50% Gray	0.50	0.50	0.50	0.2034	0.2140	0.2330	53.39	0.00	0.00
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00	0.2155	0.1111	0.0101	39.77	64.51	54.13
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00	0.0883	0.0455	0.0041	25.42	47.91	37.91
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00	0.0210	0.0108	0.0010	9.66	29.68	15.24
P	Pink	1.00	0.50	0.50	0.5276	0.3812	0.2482	68.11	48.39	22.83

with

$$L' = \tfrac{L^*+16}{116} \qquad \text{and} \tag{14.14}$$

$$f_2(c) = \begin{cases} c^3 & \text{for } c^3 > \epsilon, \\ \frac{c-16/116}{\kappa} & \text{for } c^3 \leq \epsilon, \end{cases} \tag{14.15}$$

and ϵ, κ as defined in Eqns. (14.9–14.10). The complete Java code for the CIELAB→XYZ conversion and the implementation of the associated `ColorSpace` class can be found in Progs. 14.1 and 14.2 (pp. 363–364).

14.3 CIELUV

14.3.1 CIEXYZ→CIELUV Conversion

The CIELUV component values L^*, u^*, v^* are calculated from given X, Y, Z color coordinates as follows:

$$L^* = 116 \cdot Y' - 16, \tag{14.16}$$
$$u^* = 13 \cdot L^* \cdot (u' - u'_{\mathrm{ref}}), \tag{14.17}$$
$$v^* = 13 \cdot L^* \cdot (v' - v'_{\mathrm{ref}}), \tag{14.18}$$

with Y' as defined in Eqn. (14.7) (identical to CIELAB) and

$$\begin{aligned} u' &= f_u(X,Y,Z), & u'_{\mathrm{ref}} &= f_u(X_{\mathrm{ref}},Y_{\mathrm{ref}},Z_{\mathrm{ref}}), \\ v' &= f_v(X,Y,Z), & v'_{\mathrm{ref}} &= f_v(X_{\mathrm{ref}},Y_{\mathrm{ref}},Z_{\mathrm{ref}}), \end{aligned} \tag{14.19}$$

with the correction functions

$$f_u(X,Y,Z) = \begin{cases} 0 & \text{for } X = 0, \\ \frac{4X}{X+15Y+3Z} & \text{for } X > 0, \end{cases} \tag{14.20}$$

$$f_v(X,Y,Z) = \begin{cases} 0 & \text{for } Y = 0, \\ \frac{9Y}{X+15Y+3Z} & \text{for } Y > 0. \end{cases} \tag{14.21}$$

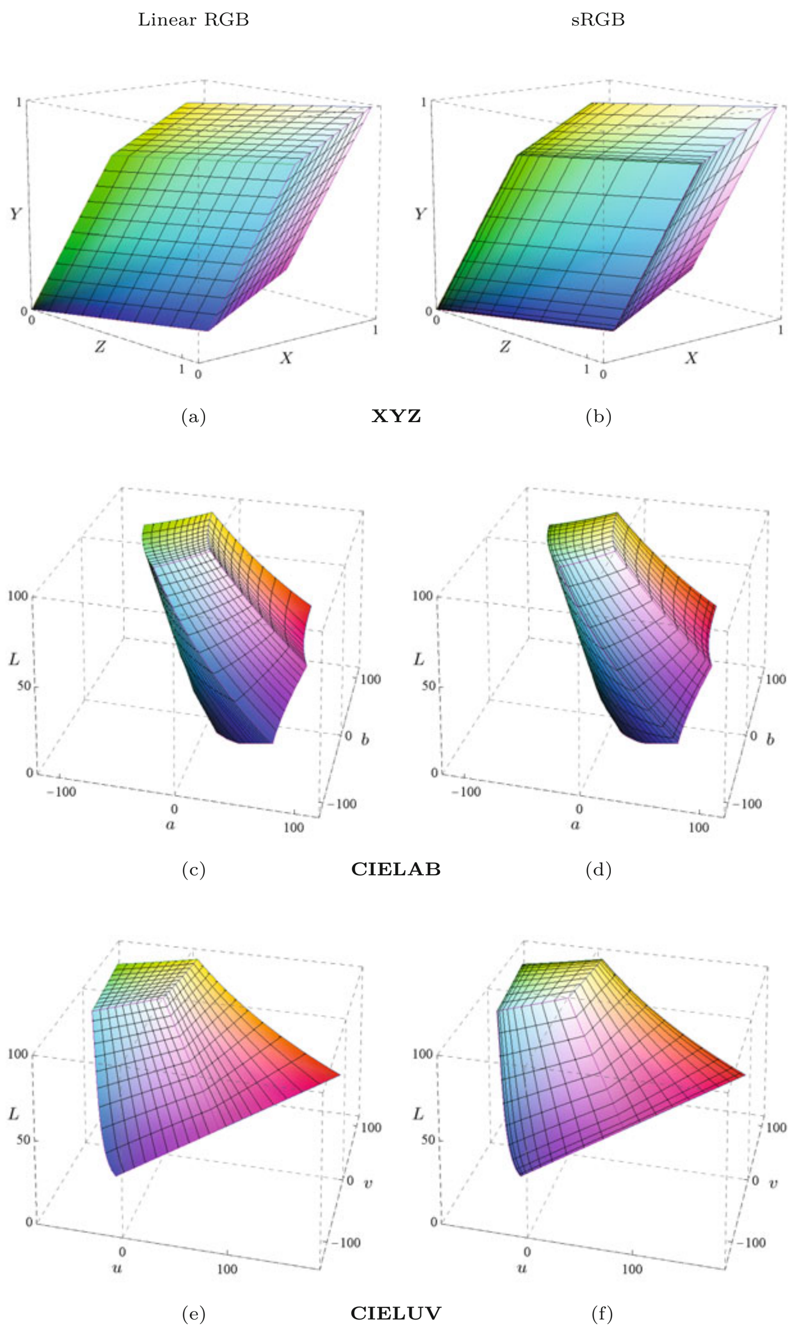


Fig. 14.5
Transformation of the RGB color cube to the XYZ, CIELAB, and CIELUV color space. The left column shows the color cube in *linear* RGB space, the right column in *nonlinear* sRGB space. Both RGB volumes were uniformly subdivided into $10 \times 10 \times 10$ cubes of equal size. In both cases, the transformation to XYZ space (a, b) yields a distorted cube with straight edges and planar faces. Due to the linear transformation from RGB to XYZ, the subdivision of the RGB cube remains uniform (a). However, the nonlinear transformation (due to gamma correction) from sRGB to XYZ makes the tesselation strongly nonuniform in XYZ space (b). Since CIELAB uses gamma correction as well, the transformation of the linear RGB cube in (c) appears much less uniform than the nonlinear sRGB cube in (d), although this appears to be the other way round in CIELUV (e, f). Note that the RGB/sRGB color cube maps to a *non-convex* volume in both the CIELAB and the CLIELUV space.

Note that the checks for zero X, Y in Eqns. (14.20)–(14.21) are not part of the original definitions but are essential in any real implementation to avoid divisions by zero.[4]

[4] Remember though that floating-point values (`double`, `float`) should never be strictly tested against zero but compared to a sufficiently small (epsilon) quantity (see Sec. F.1.8 in the Appendix).

Table 14.4
CIELUV coordinates for selected color points in sRGB. Reference white point is D65. The L^* values are identical to CIELAB (see Table 14.3).

		sRGB			CIEXYZ (D65)			CIELUV		
Pt.	Color	R'	G'	B'	X_{65}	Y_{65}	Z_{65}	L^*	u^*	v^*
S	Black	0.00	0.00	0.00	0.0000	0.0000	0.0000	0.00	0.00	0.00
R	Red	1.00	0.00	0.00	0.4125	0.2127	0.0193	53.24	175.01	37.75
Y	Yellow	1.00	1.00	0.00	0.7700	0.9278	0.1385	97.14	7.70	106.78
G	Green	0.00	1.00	0.00	0.3576	0.7152	0.1192	87.74	−83.08	107.39
C	Cyan	0.00	1.00	1.00	0.5380	0.7873	1.0694	91.11	−70.48	−15.20
B	Blue	0.00	0.00	1.00	0.1804	0.0722	0.9502	32.30	−9.40	−130.34
M	Magenta	1.00	0.00	1.00	0.5929	0.2848	0.9696	60.32	84.07	−108.68
W	White	1.00	1.00	1.00	0.9505	1.0000	1.0888	100.00	0.00	0.00
K	50% Gray	0.50	0.50	0.50	0.2034	0.2140	0.2330	53.39	0.00	0.00
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00	0.2155	0.1111	0.0101	39.77	130.73	28.20
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00	0.0883	0.0455	0.0041	25.42	83.56	18.02
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00	0.0210	0.0108	0.0010	9.66	31.74	6.85
P	Pink	1.00	0.50	0.50	0.5276	0.3812	0.2482	68.11	92.15	19.88

14.3.2 CIELUV→CIEXYZ Conversion

The reverse mapping from L^*, u^*, v^* components to X, Y, Z coordinates is defined as follows:

$$Y = Y_{\mathrm{ref}} \cdot f_2\left(\tfrac{L^*+16}{116}\right), \tag{14.22}$$

with $f_2()$ as defined in Eqn. (14.15), and

$$X = Y \cdot \frac{9u'}{4v'}, \qquad Z = Y \cdot \frac{12 - 3u' - 20v'}{4v'}, \tag{14.23}$$

with

$$(u', v') = \begin{cases} (u'_{\mathrm{ref}}, v'_{\mathrm{ref}}) & \text{for } L^* = 0, \\ (u'_{\mathrm{ref}}, v'_{\mathrm{ref}}) + \frac{1}{13 \cdot L^*} \cdot (u^*, v^*) & \text{for } L^* > 0, \end{cases} \tag{14.24}$$

and $u'_{\mathrm{ref}}, v'_{\mathrm{ref}}$ as in Eqn. (14.19).[5]

14.3.3 Measuring Color Differences

Due to its high uniformity with respect to human color perception, the CIELAB color space is a particularly good choice for determining the difference between colors (the same holds for the CIELUV space) [94, p. 57]. The difference between two color points $\mathbf{c}_1 = (L_1^*, a_1^*, b_1^*)$ and $\mathbf{c}_2 = (L_2^*, a_2^*, b_2^*)$ can be found by simply measuring the *Euclidean distance* in CIELAB or CIELUV space, for example,

$$\mathrm{ColorDist}(\mathbf{c}_1, \mathbf{c}_2) = \|\mathbf{c}_1 - \mathbf{c}_2\| \tag{14.25}$$

$$= \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2}. \tag{14.26}$$

14.4 Standard RGB (sRGB)

CIE-based color spaces such as CIELAB (and CIELUV) are device-independent and have a gamut sufficiently large to represent virtually

[5] No explicit check for zero denominators is required in Eqn. (14.23) since v' can be assumed to be greater than zero.

all visible colors in the CIEXYZ system. However, in many computer-based, display-oriented applications, such as computer graphics or multimedia, the direct use of CIE-based color spaces may be too cumbersome or inefficient.

sRGB ("standard RGB" [119]) was developed (jointly by Hewlett-Packard and Microsoft) with the goal of creating a precisely specified color space for these applications, based on standardized mappings with respect to the colorimetric CIEXYZ color space. This includes precise specifications of the three primary colors, the white reference point, ambient lighting conditions, and gamma values. Interestingly, the sRGB color specification is the same as the one specified many years before for the European PAL/SECAM television standards. Compared to CIELAB, sRGB exhibits a relatively small gamut (see Fig. 14.3), which, however, includes most colors that can be reproduced by current computer and video monitors. Although sRGB was not designed as a universal color space, its CIE-based specification at least permits more or less exact conversions to and from other color spaces.

Several standard image formats, including EXIF (JPEG) and PNG are based on sRGB color data, which makes sRGB the de facto standard for digital still cameras, color printers, and other imaging devices at the consumer level [107]. sRGB is used as a relatively dependable archive format for digital images, particularly in less demanding applications that do not require (or allow) explicit color management [225]. Thus, in practice, working with any RGB color data almost always means dealing with sRGB. It is thus no coincidence that sRGB is also the common color scheme in Java and is extensively supported by the Java standard API (see Sec. 14.7 for details).

Table 14.5 lists the key parameters of the sRGB color space (i.e., the XYZ coordinates for the primary colors **R**, **G**, **B** and the white point **W** (D65)), which are defined according to ITU-R BT.709 [122] (see Tables 14.1 and 14.2). Together, these values permit the unambiguous mapping of all other colors in the CIE diagram.

Pt.	R	G	B	X_{65}	Y_{65}	Z_{65}	x_{65}	y_{65}
R	1.0	0.0	0.0	0.412453	0.212671	0.019334	0.6400	0.3300
G	0.0	1.0	0.0	0.357580	0.715160	0.119193	0.3000	0.6000
B	0.0	0.0	1.0	0.180423	0.072169	0.950227	0.1500	0.0600
W	1.0	1.0	1.0	0.950456	1.000000	1.088754	0.3127	0.3290

Table 14.5
sRGB tristimulus values **R**, **G**, **B** with reference to the white point D65 (**W**).

14.4.1 Linear vs. Nonlinear Color Components

sRGB is a *nonlinear* color space with respect to the XYZ coordinate system, and it is important to carefully distinguish between the *linear* and *nonlinear* RGB component values. The nonlinear values (denoted R', G', B') represent the actual color tuples, the data values read from an image file or received from a digital camera. These values are pre-corrected with a fixed Gamma (≈ 2.2) such that they can be easily viewed on a common color monitor without any additional conversion. The corresponding *linear* components (denoted

R, G, B) relate to the CIEXYZ color space by a linear mapping and can thus be computed from X, Y, Z coordinates and vice versa by simple matrix multiplication, that is,

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \boldsymbol{M}_{\text{RGB}} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \boldsymbol{M}_{\text{RGB}}^{-1} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{14.27}$$

with

$$\boldsymbol{M}_{\text{RGB}} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix}, \tag{14.28}$$

$$\boldsymbol{M}_{\text{RGB}}^{-1} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix}. \tag{14.29}$$

Notice that the three column vectors of $\boldsymbol{M}_{\text{RGB}}^{-1}$ (Eqn. (14.29)) are the coordinates of the primary colors **R**, **G**, **B** (tristimulus values) in XYZ space (cf. Table 14.5) and thus

$$\mathbf{R} = \boldsymbol{M}_{\text{RGB}}^{-1} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{G} = \boldsymbol{M}_{\text{RGB}}^{-1} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{B} = \boldsymbol{M}_{\text{RGB}}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{14.30}$$

14.4.2 CIEXYZ→sRGB Conversion

To transform a given XYZ color to sRGB (Fig. 14.6), we first compute the *linear* R, G, B values by multiplying the (X, Y, Z) coordinate vector with the matrix $\boldsymbol{M}_{\text{RGB}}$ (Eqn. (14.28)),

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \boldsymbol{M}_{\text{RGB}} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \tag{14.31}$$

Subsequently, a modified gamma correction (see Ch. 4, Sec. 4.7.6) with $\gamma = 2.4$ (which corresponds to an effective gamma value of ca. 2.2) is applied to the linear R, G, B values,

$$R' = f_1(R), \qquad G' = f_1(G), \qquad B' = f_1(B), \tag{14.32}$$

with

$$f_1(c) = \begin{cases} 12.92 \cdot c & \text{for } c \leq 0.0031308, \\ 1.055 \cdot c^{1/2.4} - 0.055 & \text{for } c > 0.0031308. \end{cases} \tag{14.33}$$

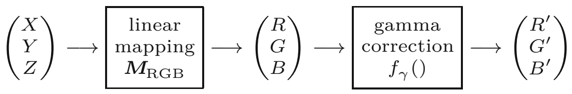

X Y Z
linear mapping $\boldsymbol{M}_{\text{RGB}}$
R G B
gamma correction $f_\gamma()$
R′ G′ B′

Fig. 14.6
Color transformation from CIEXYZ to sRGB.

The resulting sRGB components R', G', B' are limited to the interval $[0, 1]$ (see Table 14.6). To obtain discrete numbers, the R', G', B' values are finally scaled linearly to the 8-bit integer range $[0, 255]$.

		sRGB (*nonlinear*)			RGB (*linear*)			CIEXYZ		
Pt.	Color	R'	G'	B'	R	G	B	X_{65}	Y_{65}	Z_{65}
S	Black	0.00	0.00	0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
R	Red	1.00	0.00	0.00	1.0000	0.0000	0.0000	0.4125	0.2127	0.0193
Y	Yellow	1.00	1.00	0.00	1.0000	1.0000	0.0000	0.7700	0.9278	0.1385
G	Green	0.00	1.00	0.00	0.0000	1.0000	0.0000	0.3576	0.7152	0.1192
C	Cyan	0.00	1.00	1.00	0.0000	1.0000	1.0000	0.5380	0.7873	1.0694
B	Blue	0.00	0.00	1.00	0.0000	0.0000	1.0000	0.1804	0.0722	0.9502
M	Magenta	1.00	0.00	1.00	1.0000	0.0000	1.0000	0.5929	0.2848	0.9696
W	White	1.00	1.00	1.00	1.0000	1.0000	1.0000	0.9505	1.0000	1.0888
K	50% Gray	0.50	0.50	0.50	0.2140	0.2140	0.2140	0.2034	0.2140	0.2330
$\mathbf{R}_{75}$	75% Red	0.75	0.00	0.00	0.5225	0.0000	0.0000	0.2155	0.1111	0.0101
$\mathbf{R}_{50}$	50% Red	0.50	0.00	0.00	0.2140	0.0000	0.0000	0.0883	0.0455	0.0041
$\mathbf{R}_{25}$	25% Red	0.25	0.00	0.00	0.0509	0.0000	0.0000	0.0210	0.0108	0.0010
P	Pink	1.00	0.50	0.50	1.0000	0.2140	0.2140	0.5276	0.3812	0.2482

Table 14.6
CIEXYZ coordinates for selected sRGB colors. The table lists the *nonlinear* R', G', and B' components, the *linearized* R, G, and B values, and the corresponding X, Y, and Z coordinates (for white point D65). The linear and nonlinear RGB values are identical for the extremal points of the RGB color cube $\mathbf{S},\ldots,\mathbf{W}$ (top rows) because the gamma correction does not affect 0 and 1 component values. However, *intermediate* colors ($\mathbf{K},\ldots,\mathbf{P}$, shaded rows) may exhibit large differences between the nonlinear and linear components (e.g., compare the R' and R values for $\mathbf{R}_{25}$).

14.4.3 sRGB→CIEXYZ Conversion

To calculate the reverse transformation from sRGB to XYZ, the given (nonlinear) $R'G'B'$ values (in the range $[0,1]$) are first linearized by inverting the gamma correction (Eqn. (14.33)), that is,

$$R = f_2(R'), \quad G = f_2(G'), \quad B = f_2(B'), \tag{14.34}$$

with

$$f_2(c') = \begin{cases} \frac{c'}{12.92} & \text{for } c' \le 0.04045, \\ \left(\frac{c'+0.055}{1.055}\right)^{2.4} & \text{for } c' > 0.04045. \end{cases} \tag{14.35}$$

Subsequently, the linearized (R,G,B) vector is transformed to XYZ coordinates by multiplication with the inverse of the matrix $\boldsymbol{M}_{\mathrm{RGB}}$ (Eqn. (14.29)),

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \boldsymbol{M}_{\mathrm{RGB}}^{-1} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{14.36}$$

14.4.4 Calculations with Nonlinear sRGB Values

Due to the wide use of sRGB in digital photography, graphics, multimedia, Internet imaging, etc., there is a probability that a given image is encoded in sRGB colors. If, for example, a JPEG image is opened with ImageJ or Java, the pixel values in the resulting data array are media-oriented (i.e., nonlinear R', G', B' components of the sRGB color space). Unfortunately, this fact is often overlooked by programmers, with the consequence that colors are incorrectly manipulated and reproduced.

As a general rule, any arithmetic operation on color values should always be performed on the *linearized* R, G, B components, which are obtained from the nonlinear R', G', B' values through the inverse gamma function f_γ^{-1} (Eqn. (14.35)) and converted back again with f_γ (Eqn. (14.33)).

Example: color to grayscale conversion

The principle of converting RGB colors to grayscale values by computing a weighted sum of the color components was described already in Chapter 12, Sec. 12.2.1, where we had simply ignored the issue of possible nonlinearities. As one may have guessed, however, the variables R, G, B, and Y in Eqn. (12.10) on p. 305,

$$Y = 0.2125 \cdot R + 0.7154 \cdot G + 0.072 \cdot B \tag{14.37}$$

implicitly refer to *linear* color and gray values, respectively, and not the raw sRGB values! Based on Eqn. (14.37), the *correct* grayscale conversion from raw (nonlinear) sRGB components R', G', B' is

$$Y' = f_1\big(0.2125 \cdot f_2(R') + 0.7154 \cdot f_2(G') + 0.0721 \cdot f_2(B')\big), \tag{14.38}$$

with $f_\gamma()$ and $f_\gamma^{-1}()$ as defined in Eqns. (14.33) and (14.35). The result (Y') is again a nonlinear, sRGB-compatible gray value; that is, the sRGB color tuple (Y', Y', Y') should have the same perceived luminance as the original color (R', G', B').

Note that setting the components of an sRGB color pixel to three arbitrary but identical values Y',

$$(R', G', B') \leftarrow (Y', Y', Y')$$

always creates a gray (colorless) pixel, despite the nonlinearities of the sRGB space. This is due to the fact that the gamma correction (Eqns. (14.33) and (14.35)) applies evenly to all three color components and thus any three identical values map to a (linearized) color on the straight gray line between the black point $\mathbf{S}$ and the white point $\mathbf{W}$ in XYZ space (cf. Fig. 14.1(b)).

For many applications, however, the following *approximation* to the exact grayscale conversion in Eqn. (14.38) is sufficient. It works without converting the sRGB values (i.e., directly on the nonlinear R', G', B' components) by computing a linear combination

$$Y' \approx w'_R \cdot R' + w'_G \cdot G' + w'_B \cdot B', \tag{14.39}$$

with a slightly different set of weights; for example, $w'_R = 0.309$, $w'_G = 0.609$, $w'_B = 0.082$, as proposed in [188]. The resulting quantity from Eqn. (14.39) is sometimes called *luma* (compared to *luminance* in Eqn. (14.37)).

14.5 Adobe RGB

A distinct weakness of sRGB is its relatively small gamut, which is limited to the range of colors reproducible by ordinary color monitors. This causes problems, for example, in printing, where larger gamuts are needed, particularly in the green regions. The "Adobe RGB (1998)" [1] color space, developed by Adobe as their own standard, is based on the same general concept as sRGB but exhibits a significantly larger gamut (Fig. 14.3), which extends its use particularly to print applications. Figure 14.7 shows the noted difference between the sRGB and Adobe RGB gamuts in 3D CIEXYZ color space.

The neutral point of Adobe RGB corresponds to the D65 standard (with $x = 0.3127$, $y = 0.3290$), and the gamma value is 2.199

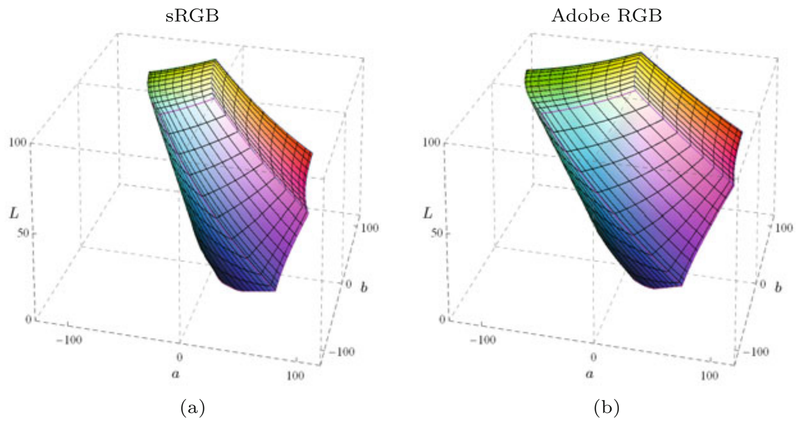


Fig. 14.7
Gamuts of sRGB and Adobe RGB shown in CIELAB color space. The volume of the sRGB gamut (a) is significantly smaller than the Adobe RGB gamut (b), particularly in the green color region. The tesselation corresponds to a uniform subdivision of the original RGB cubes (in the respective color spaces).

(compared with 2.4 for sRGB) for the forward correction and $\frac{1}{2.199}$ for the inverse correction, respectively. The associated file specification provides for a number of different codings (8- to 16-bit integer and 32-bit floating point) for the color components. Adobe RGB is frequently used in professional photography as an alternative to the CIELAB color space and for picture archive applications.

14.6 Chromatic Adaptation

The human eye has the capability to interpret colors as being constant under varying viewing conditions and illumination in particular. A white sheet of paper appears white to us in bright daylight as well as under fluorescent lighting, although the spectral composition of the light that enters the eye is completely different in both situations. The CIE color system takes into account the color temperature of the ambient lighting because the exact interpretation of XYZ color values also requires knowledge of the corresponding reference white point. For example, a color value (X, Y, Z) specified with respect to the D50 reference white point is generally perceived differently when reproduced by a D65-based media device, although the absolute (i.e., measured) color is the same. Thus the actual meaning of XYZ values cannot be known without knowing the corresponding white point. This is known as *relative colorimetry*.

If colors are specified with respect to *different* white points, for example $\mathbf{W}_1 = (X_{W1}, Y_{W1}, Z_{W1})$ and $\mathbf{W}_2 = (X_{W2}, Y_{W2}, Z_{W2})$, they can be related by first applying a so-called *chromatic adaptation transformation* (CAT) [114, Ch. 34] in XYZ color space. This transformation determines, for given color coordinates (X_1, Y_1, Z_1) and the associated white point $\mathbf{W}_1$, the new color coordinates (X_2, Y_2, Z_2) relative to another white point $\mathbf{W}_2$.

14.6.1 XYZ Scaling

The simplest chromatic adaptation method is XYZ scaling, where the individual color coordinates are individually multiplied by the ratios of the corresponding white point coordinates, that is,

$$X_2 = X_1 \cdot \frac{\hat{X}_2}{\hat{X}_1}, \qquad Y_2 = Y_1 \cdot \frac{\hat{Y}_2}{\hat{Y}_1}, \qquad Z_2 = Z_1 \cdot \frac{\hat{Z}_2}{\hat{Z}_1}. \tag{14.40}$$

For example, for converting colors (X_{65}, Y_{65}, Z_{65}) related to the white point $\mathbf{D65} = (\hat{X}_{65}, \hat{Y}_{65}, \hat{Z}_{65})$ to the corresponding colors for white point $\mathbf{D50} = (\hat{X}_{50}, \hat{Y}_{50}, \hat{Z}_{50})$,[6] the concrete scaling is

$$\begin{aligned} X_{50} &= X_{65} \cdot \tfrac{\hat{X}_{50}}{\hat{X}_{65}} = X_{65} \cdot \tfrac{0.964296}{0.950456} = X_{65} \cdot 1.01456, \\ Y_{50} &= Y_{65} \cdot \tfrac{\hat{Y}_{50}}{\hat{Y}_{65}} = Y_{65} \cdot \tfrac{1.000000}{1.000000} = Y_{65}, \\ Z_{50} &= Z_{65} \cdot \tfrac{\hat{Z}_{50}}{\hat{Z}_{65}} = Z_{65} \cdot \tfrac{0.825105}{1.088754} = Z_{65} \cdot 0.757843\,. \end{aligned} \tag{14.41}$$

This form of scaling the color coordinates in XYZ space is usually not considered a good color adaptation model and is not recommended for high-quality applications.

14.6.2 Bradford Adaptation

The most common chromatic adaptation models are based on scaling the color coordinates not directly in XYZ but in a “virtual” $R^*G^*B^*$ color space obtained from the XYZ values by a linear transformation

$$\begin{pmatrix} R^* \\ G^* \\ B^* \end{pmatrix} = \boldsymbol{M}_{\mathrm{CAT}} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{14.42}$$

where $\boldsymbol{M}_{\mathrm{CAT}}$ is a 3×3 transformation matrix (defined in Eqn. (14.45)). After appropriate scaling, the $R^*G^*B^*$ coordinates are transformed back to XYZ, so the complete adaptation transform from color coordinates X_1, Y_1, Z_1 (w.r.t. white point $\mathbf{W}_1 = (X_{\mathrm{W1}}, Y_{\mathrm{W1}}, Z_{\mathrm{W1}})$) to the new color coordinates X_2, Y_2, Z_2 (w.r.t. white point $\mathbf{W}_2 = (X_{\mathrm{W2}}, Y_{\mathrm{W2}}, Z_{\mathrm{W2}})$) takes the form

$$\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} = \boldsymbol{M}_{\mathrm{CAT}}^{-1} \cdot \begin{pmatrix} \frac{R^*_{\mathrm{W2}}}{R^*_{\mathrm{W1}}} & 0 & 0 \\ 0 & \frac{G^*_{\mathrm{W2}}}{G^*_{\mathrm{W1}}} & 0 \\ 0 & 0 & \frac{B^*_{\mathrm{W2}}}{B^*_{\mathrm{W1}}} \end{pmatrix} \cdot \boldsymbol{M}_{\mathrm{CAT}} \cdot \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix}, \tag{14.43}$$

where the diagonal elements $\frac{R^*_{\mathrm{W2}}}{R^*_{\mathrm{W1}}}, \frac{G^*_{\mathrm{W2}}}{G^*_{\mathrm{W1}}}, \frac{B^*_{\mathrm{W2}}}{B^*_{\mathrm{W1}}}$ are the (constant) ratios of the $R^*G^*B^*$ values of the white points $\mathbf{W}_2$, $\mathbf{W}_1$, respectively; that is,

$$\begin{pmatrix} R^*_{\mathrm{W1}} \\ G^*_{\mathrm{W1}} \\ B^*_{\mathrm{W1}} \end{pmatrix} = \boldsymbol{M}_{\mathrm{CAT}} \cdot \begin{pmatrix} X_{\mathrm{W1}} \\ Y_{\mathrm{W1}} \\ Z_{\mathrm{W1}} \end{pmatrix}, \quad \begin{pmatrix} R^*_{\mathrm{W2}} \\ G^*_{\mathrm{W2}} \\ B^*_{\mathrm{W2}} \end{pmatrix} = \boldsymbol{M}_{\mathrm{CAT}} \cdot \begin{pmatrix} X_{\mathrm{W2}} \\ Y_{\mathrm{W2}} \\ Z_{\mathrm{W2}} \end{pmatrix}. \tag{14.44}$$

The “Bradford” model [114, p. 590] specifies for Eqn. (14.43) the particular transformation matrix

$$\boldsymbol{M}_{\mathrm{CAT}} = \begin{pmatrix} 0.8951 & 0.2664 & -0.1614 \\ -0.7502 & 1.7135 & 0.0367 \\ 0.0389 & -0.0685 & 1.0296 \end{pmatrix}. \tag{14.45}$$

[6] See Table 14.2.

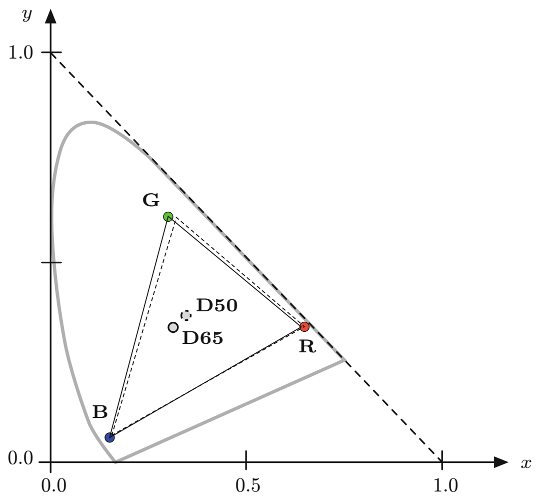


Fig. 14.8
Bradford chromatic adaptation from white point D65 to D50. The solid triangle represents the original RGB gamut for white point D65, with the primaries $(\mathbf{R}, \mathbf{G}, \mathbf{B})$ located at the corner points. The dashed triangle is the corresponding gamut after chromatic adaptation to white point D50.

Inserting $\boldsymbol{M}_{\mathrm{CAT}}$ matrix in Eqn. (14.43) gives the complete chromatic adaptation. For example, the resulting transformation for converting from D65-based to D50-based colors (i.e., $\mathbf{W}_1 = \mathbf{D65}$, $\mathbf{W}_2 = \mathbf{D50}$, as listed in Table 14.2) is

$$\begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix} = \boldsymbol{M}_{50|65} \cdot \begin{pmatrix} X_{65} \\ Y_{65} \\ Z_{65} \end{pmatrix} = \begin{pmatrix} 1.047884 & 0.022928 & -0.050149 \\ 0.029603 & 0.990437 & -0.017059 \\ -0.009235 & 0.015042 & 0.752085 \end{pmatrix} \cdot \begin{pmatrix} X_{65} \\ Y_{65} \\ Z_{65} \end{pmatrix}, \quad (14.46)$$

and conversely from D50-based to D65-based colors (i.e., $\mathbf{W}_1 = \mathbf{D50}$, $\mathbf{W}_2 = \mathbf{D65}$),

$$\begin{pmatrix} X_{65} \\ Y_{65} \\ Z_{65} \end{pmatrix} = \boldsymbol{M}_{65|50} \cdot \begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix} = \boldsymbol{M}_{50|65}^{-1} \cdot \begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix} = \begin{pmatrix} 0.955513 & -0.023079 & 0.063190 \\ -0.028348 & 1.009992 & 0.021019 \\ 0.012300 & -0.020484 & 1.329993 \end{pmatrix} \cdot \begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix}. \quad (14.47)$$

Figure 14.8 illustrates the effects of adaptation from the D65 white point to D50 in the CIE x, y chromaticity diagram. A short list of corresponding color coordinates is given in Table 14.7.

The Bradford model is a widely used chromatic adaptation scheme but several similar procedures have been proposed (see also Exercise 14.1). Generally speaking, chromatic adaptation and related problems have a long history in color engineering and are still active fields of scientific research [258, Ch. 5, Sec. 5.12].

Table 14.7
Bradford chromatic adaptation from white point D65 to D50 for selected sRGB colors. The XYZ coordinates X_{65}, Y_{65}, Z_{65} relate to the original white point D65 ($\mathbf{W}_1$). X_{50}, Y_{50}, Z_{50} are the corresponding coordinates for the new white point D50 ($\mathbf{W}_2$), obtained with the Bradford adaptation according to Eqn. (14.46).

		sRGB			XYZ (**D65**)			XYZ (**D50**)		
Pt.	Color	R'	G'	B'	X_{65}	Y_{65}	Z_{65}	X_{50}	Y_{50}	Z_{50}
S	Black	0.00	0.0	0.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
R	Red	1.00	0.0	0.0	0.4125	0.2127	0.0193	0.4361	0.2225	0.0139
Y	Yellow	1.00	1.0	0.0	0.7700	0.9278	0.1385	0.8212	0.9394	0.1110
G	Green	0.00	1.0	0.0	0.3576	0.7152	0.1192	0.3851	0.7169	0.0971
C	Cyan	0.00	1.0	1.0	0.5380	0.7873	1.0694	0.5282	0.7775	0.8112
B	Blue	0.00	0.0	1.0	0.1804	0.0722	0.9502	0.1431	0.0606	0.7141
M	Magenta	1.00	0.0	1.0	0.5929	0.2848	0.9696	0.5792	0.2831	0.7280
W	White	1.00	1.0	1.0	0.9505	1.0000	1.0888	0.9643	1.0000	0.8251
K	50% Gray	0.50	0.5	0.5	0.2034	0.2140	0.2330	0.2064	0.2140	0.1766
$\mathbf{R}_{75}$	75% Red	0.75	0.0	0.0	0.2155	0.1111	0.0101	0.2279	0.1163	0.0073
$\mathbf{R}_{50}$	50% Red	0.50	0.0	0.0	0.0883	0.0455	0.0041	0.0933	0.0476	0.0030
$\mathbf{R}_{25}$	25% Red	0.25	0.0	0.0	0.0210	0.0108	0.0010	0.0222	0.0113	0.0007
P	Pink	1.00	0.5	0.5	0.5276	0.3812	0.2482	0.5492	0.3889	0.1876

14.7 Colorimetric Support in Java

sRGB is the standard color space in Java; that is, the components of color objects and RGB color images are gamma-corrected, *nonlinear* R', G', B' values (see Fig. 14.6). The nonlinear R', G', B' values are related to the linear R, G, B values by a modified gamma correction, as specified by the sRGB standard (Eqns. (14.33) and (14.35)).

14.7.1 Profile Connection Space (PCS)

The Java API (AWT) provides classes for representing color objects and color spaces, together with a rich set of corresponding methods. Java's color system is designed after the ICC[7] "color management architecture", which uses a CIEXYZ-based device-independent color space called the "profile connection space" (PCS) [118, 121]. The PCS color space is used as the intermediate reference for converting colors between different color spaces. The ICC standard defines device profiles (see Sec. 14.7.4) that specify the transforms to convert between a device's color space and the PCS. The advantage of this approach is that for any given device only a single color transformation (profile) must be specified to convert between device-specific colors and the unified, colorimetric profile connection space. Every `ColorSpace` class (or subclass) provides the methods `fromCIEXYZ()` and `toCIEXYZ()` to convert device color values to XYZ coordinates in the standardized PCS. Figure 14.9 illustrates the principal application of `ColorSpace` objects for converting colors between different color spaces in Java using the XYZ space as a common "hub".

Different to the sRGB specification, the ICC specifies **D50** (and *not* D65) as the illuminant white point for its default PCS color space (see Table 14.2). The reason is that the ICC standard was developed primarily for color management in photography, graphics, and printing, where D50 is normally used as the reflective media white point. The Java methods `fromCIEXYZ()` and `toCIEXYZ()` thus take and return X, Y, Z color coordinates that are relative to the D50 white point. The resulting coordinates for the primary colors (listed in Table 14.8) are different from the ones given for white point D65 (see Table 14.5)! This is a frequent cause of confusion since the sRGB

[7] International Color Consortium (ICC, www.color.org).

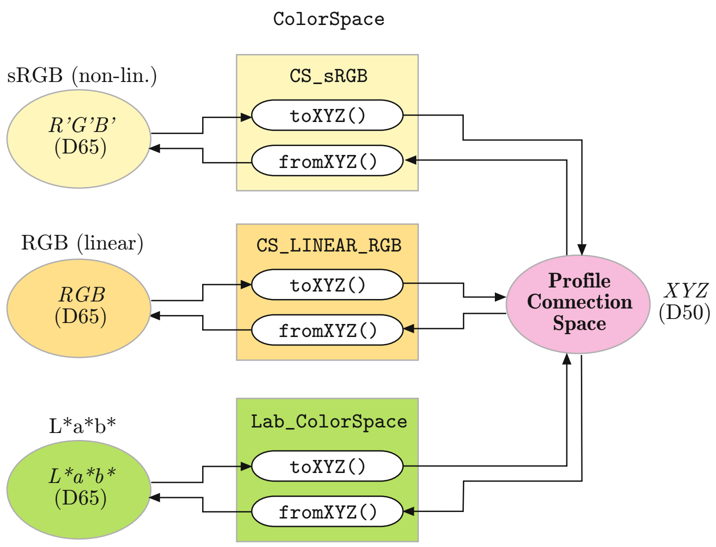


Fig. 14.9
XYZ-based color conversion in Java. `ColorSpace` objects implement the methods `fromCIEXYZ()` and `toCIEXYZ()` to convert color vectors from and to the CIEXYZ color space, respectively. Colorimetric transformations between color spaces can be accomplished as a two-step process via the XYZ space. For example, to convert from sRGB to CIELAB, the sRGB color is first converted to XYZ and subsequently from XYZ to CIELAB. Notice that Java's standard XYZ color space is based on the D50 white point, while most common color spaces refer to D65.

Pt.	R	G	B	X_{50}	Y_{50}	Z_{50}	x_{50}	y_{50}
R	1.0	0.0	0.0	0.436108	0.222517	0.013931	0.6484	0.3309
G	0.0	1.0	0.0	0.385120	0.716873	0.097099	0.3212	0.5978
B	0.0	0.0	1.0	0.143064	0.060610	0.714075	0.1559	0.0660
W	1.0	1.0	1.0	0.964296	1.000000	0.825106	0.3457	0.3585

Table 14.8
Color coordinates for sRGB primaries and the white point in Java's default XYZ color space. Color coordinates for sRGB primaries and the white point in Java's default XYZ color space. The white point **W** is equal to D50.

component values are D65-based (as specified by the sRGB standard) but Java's XYZ values are relative to the D50.

Chromatic adaptation (see Sec. 14.6) is used to convert between XYZ color coordinates that are measured with respect to different white points. The ICC specification [118] recommends a linear chromatic adaptation based on the Bradford model to convert between the D65-related XYZ coordinates (X_{65}, Y_{65}, Z_{65}) and D50-related values (X_{50}, Y_{50}, Z_{50}). This is also implemented by the Java API.

The complete mapping between the linearized sRGB color values (R, G, B) and the D50-based (X_{50}, Y_{50}, Z_{50}) coordinates can be expressed as a linear transformation composed of the RGB→XYZ$_{65}$ transformation by matrix $\boldsymbol{M}_{\mathrm{RGB}}$ (Eqns. (14.28) and (14.29)) and the chromatic adaptation transformation XYZ$_{65}$→XYZ$_{50}$ defined by the matrix $\boldsymbol{M}_{50|65}$ (Eqn. (14.46)),

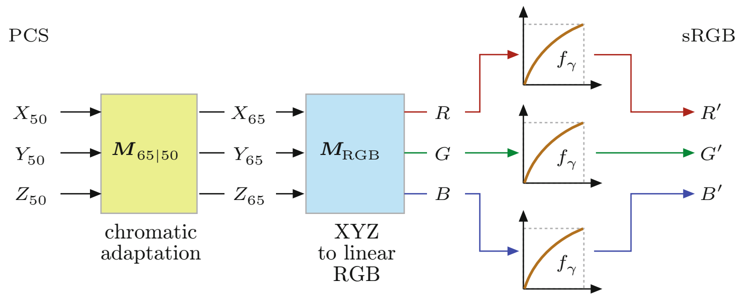
$$\begin{aligned}\begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix} &= \boldsymbol{M}_{50|65} \cdot \boldsymbol{M}_{\mathrm{RGB}}^{-1} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \left(\boldsymbol{M}_{\mathrm{RGB}} \cdot \boldsymbol{M}_{65|50}\right)^{-1} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \\ &= \begin{pmatrix} 0.436131 & 0.385147 & 0.143033 \\ 0.222527 & 0.716878 & 0.060600 \\ 0.013926 & 0.097080 & 0.713871 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \end{aligned} \tag{14.48}$$

and, in the reverse direction,

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \boldsymbol{M}_{\mathrm{RGB}} \cdot \boldsymbol{M}_{65|50} \cdot \begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix}$$
$$= \begin{pmatrix} 3.133660 & -1.617140 & -0.490588 \\ -0.978808 & 1.916280 & 0.033444 \\ 0.071979 & -0.229051 & 1.405840 \end{pmatrix} \cdot \begin{pmatrix} X_{50} \\ Y_{50} \\ Z_{50} \end{pmatrix}. \qquad (14.49)$$

Equations (14.48) and (14.49) are the transformations implemented by the methods `toCIEXYZ()` and `fromCIEXYZ()`, respectively, for Java's default sRGB `ColorSpace` class. Of course, these methods must also perform the necessary gamma correction between the linear R, G, B components and the actual (nonlinear) sRGB values R', G', B'. Figure 14.10 illustrates the complete transformation from D50-based PCS coordinates to nonlinear sRGB values.

Fig. 14.10
Transformation from **D50**-based CIEXYZ coordinates (X_{50}, Y_{50}, Z_{50}) in Java's *Profile Connection Space* (PCS) to nonlinear sRGB values (R', G', B'). The first step ist chromatic adaptation from **D50** to **D65** (by $\boldsymbol{M}_{65|50}$), followed by mapping the CIEXYZ coordinates to linear RGB values (by $\boldsymbol{M}_{\mathrm{RGB}}$). Finally, gamma correction is applied individually to all three color components.



14.7.2 Color-Related Java Classes

The Java standard API offers extensive support for working with colors and color images. The most important classes contained in the Java AWT package are:

- `Color`: defines individual color objects.
- `ColorSpace`: specifies entire color spaces.
- `ColorModel`: describes the structure of color images; e.g., full-color images or indexed-color images (see Prog. 12.3 on p. 301).

Class `Color` *(*`java.awt.Color`*)*

An object of class `Color` describes a particular color in the associated color space, which defines the number and type of the color components. `Color` objects are primarily used for graphic operations, such as to specify the color for drawing or filling graphic objects. Unless the color space is not explicitly specified, new `Color` objects are created as sRGB colors. The arguments passed to the `Color` constructor methods may be either `float` components in the range $[0, 1]$ or integers in the range $[0, 255]$, as demonstrated by the following example:

```
Color pink = new Color(1.0f, 0.5f, 0.5f);
Color blue = new Color(0, 0, 255);
```

Note that in both cases the arguments are interpreted as *nonlinear* sRGB values (R', G', B'). Other constructor methods exist for class

Color that also accept alpha (transparency) values. In addition, the Color class offers two useful static methods, RGBtoHSB() and HSBtoRGB(), for converting between sRGB and HSV[8] colors (see Ch. 12, Sec. 12.2.3).

Class ColorSpace (java.awt.color.ColorSpace)

An object of type ColorSpace represents an entire color space, such as sRGB or CMYK. Every subclass of ColorSpace (which itself is an abstract class) provides methods for converting its native colors to the CIEXYZ and sRGB color space and vice versa, such that conversions between arbitrary color spaces can easily be performed (through Java's XYZ-based profile connection space). In the following example, we first create an instance of the default sRGB color space by invoking the static method ColorSpace.getInstance() and subsequently convert an sRGB color object (R', B', G') to the corresponding (X, Y, Z) coordinates in Java's (D50-based) profile connection space:

```
// create an sRGB color space object:
  ColorSpace sRGBcsp
        = ColorSpace.getInstance(ColorSpace.CS_sRGB);
  float[] pink_RGB = new float[] {1.0f, 0.5f, 0.5f};
  // convert from sRGB to XYZ:
  float[] pink_XYZ = sRGBcsp.toCIEXYZ(pink_RGB);
```

Notice that color vectors are represented as float[] arrays for color conversions with ColorSpace objects. If required, the method getComponents() can be used to convert Color objects to float[] arrays. In summary, the types of color spaces that can be created with the ColorSpace.getInstance() method include:

- CS_sRGB: the standard (D65-based) RGB color space with *non-linear* R', G', B' components, as specified in [119].
- CS_LINEAR_RGB: color space with *linear* R, G, B components (i.e., no gamma correction applied).
- CS_GRAY: single-component color space with linear grayscale values.
- CS_PYCC: Kodak's Photo YCC color space.
- CS_CIEXYZ: the default XYZ profile connection space (based on the D50 white point).

Other color spaces can be implemented by creating additional implementations (subclasses) of ColorSpace, as demonstrated for CIELAB in the example in Sec. 14.7.3.

14.7.3 Implementation of the CIELAB Color Space (Example)

In the following, we show a complete implementation of the CIELAB color space, which is not available in the current Java API, based on the specification given in Sec. 14.2. For this purpose, we define a

[8] The HSV color space is referred to as "HSB" (hue, saturation, *brightness*) in the Java API.

subclass of ColorSpace (defined in the package java.awt.color) named Lab_ColorSpace, which implements the required methods toCIEXYZ(), fromCIEXYZ() for converting to and from Java's default profile connection space, respectively, and toRGB(), fromRGB() for converting between CIELAB and sRGB (Progs. 14.1 and 14.2). These conversions are performed in two steps via XYZ coordinates, where care must be taken regarding the right choice of the associated white point (CIELAB is based on D65 and Java XYZ on D50). The following examples demonstrate the principal use of the new Lab_ColorSpace class:[9]

```
ColorSpace labCs = new LabColorSpace();
float[] cyan_sRGB = {0.0f, 1.0f, 1.0f};
float[] cyan_LAB = labCs.fromRGB(cyan_sRGB) // sRGB→LAB
float[] cyan_XYZ = labCs.toXYZ(cyan_LAB);   // LAB→XYZ (D50)
```

14.7.4 ICC Profiles

Even with the most precise specification, a standard color space may not be sufficient to accurately describe the transfer characteristics of some input or output device. ICC[10] profiles are standardized descriptions of individual device transfer properties that warrant that an image or graphics can be reproduced accurately on different media. The contents and the format of ICC profile files is specified in [118], which is identical to ISO standard 15076 [121]. Profiles are thus a key element in the process of digital color management [246].

The Java graphics API supports the use of ICC profiles mainly through the classes ICC_ColorSpace and ICC_Profile, which allow application designers to create various standard profiles and read ICC profiles from data files.

Assume, for example, that an image was recorded with a calibrated scanner and shall be displayed accurately on a monitor. For this purpose, we need the ICC profiles for the scanner and the monitor, which are often supplied by the manufacturers as .icc data files.[11] For standard color spaces, the associated ICC profiles are often available as part of the computer installation, such as CIERGB.icc or NTSC1953.icc. With these profiles, a color space object can be specified that converts the image data produced by the scanner into corresponding CIEXYZ or sRGB values, as illustrated by the following example:

```
// load the scanner's ICC profile and create a corresponding color space:
ICC_ColorSpace scannerCs = new
  ICC_ColorSpace(ICC_ProfileRGB.getInstance("scanner.icc"));

// specify a device-specific color:
float[] deviceColor = {0.77f, 0.13f, 0.89f};
```

[9] Classes LabColorSpace, LuvColorSpace (analogous implementation of the CIELUV color space) and associated auxiliary classes are found in package imagingbook.pub.colorimage.

[10] International Color Consortium ICC (www.color.org).

[11] ICC profile files may also come with extensions .icm or .pf (as in the Java distribution).

```java
package imagingbook.pub.color.image;

import static imagingbook.pub.color.image.Illuminant.D50;
import static imagingbook.pub.color.image.Illuminant.D65;

import java.awt.color.ColorSpace;

public class LabColorSpace extends ColorSpace {

  // D65 reference white point and chromatic adaptation objects:
  static final double Xref = D65.X; // 0.950456
  static final double Yref = D65.Y; // 1.000000
  static final double Zref = D65.Z; // 1.088754

  static final ChromaticAdaptation catD65toD50 =
        new BradfordAdaptation(D65, D50);
  static final ChromaticAdaptation catD50toD65 =
        new BradfordAdaptation(D50, D65);

  // the only constructor:
  public LabColorSpace() {
    super(TYPE_Lab,3);
  }

  // XYZ (Profile Connection Space, D50) → CIELab conversion:
  public float[] fromCIEXYZ(float[] XYZ50) {
    float[] XYZ65 = catD50toD65.apply(XYZ50);
    return fromCIEXYZ65(XYZ65);
  }

  // XYZ (D65) → CIELab conversion (Eqn. (14.6)–14.10):
  public float[] fromCIEXYZ65(float[] XYZ65) {
    double xx = f1(XYZ65[0] / Xref);
    double yy = f1(XYZ65[1] / Yref);
    double zz = f1(XYZ65[2] / Zref);
    float L = (float)(116.0 * yy - 16.0);
    float a = (float)(500.0 * (xx - yy));
    float b = (float)(200.0 * (yy - zz));
    return new float[] {L, a, b};
  }
  // CIELab→XYZ (Profile Connection Space, D50) conversion:
  public float[] toCIEXYZ(float[] Lab) {
    float[] XYZ65 = toCIEXYZ65(Lab);
    return catD65toD50.apply(XYZ65);
  }

  // CIELab→XYZ (D65) conversion (Eqn. (14.13)–14.15):
  public float[] toCIEXYZ65(float[] Lab) {
    double ll = ( Lab[0] + 16.0 ) / 116.0;
    float Y65 = (float) (Yref * f2(ll));
    float X65 = (float) (Xref * f2(ll + Lab[1] / 500.0));
    float Z65 = (float) (Zref * f2(ll - Lab[2] / 200.0));
    return new float[] {X65, Y65, Z65};
  }
```

Prog. 14.1
Java implementation of the CIELAB color space as a sub-class of `ColorSpace` (part 1). The conversion from D50-based profile connection space XYZ coordinates to CIELAB (Eqn. (14.6)) and back is implemented by the required methods `fromCIEXYZ()` and `toCIEXYZ()`, respectively. The auxiliary methods `fromCIEXYZ65()` and `toCIEXYZ65()` are used for converting D65-based XYZ coordinates (see Eqn. (14.6)). Chromatic adaptation from D50 to D65 is performed by the objects `catD65toD50` and `catD50toD65` of type `ChromaticAdaptation`. The gamma correction functions f_1 (Eqn. (14.8)) and f_2 (Eqn. (14.15)) are implemented by the methods `f1()` and `f2()`, respectively (see Prog. 14.2).

Prog. 14.2
Java implementation of the CIELAB color space as a subclass of `ColorSpace` (part 2). The methods `fromRGB()` and `toRGB()` perform direct conversion between CIELAB and sRGB via D65-based XYZ coordinates, i.e., without conversion to Java's *Profile Connection Space*. Gamma correction (for mapping between linear RGB and sRGB component values) is implemented by the methods `gammaFwd()` and `gammaInv()` in class `sRgbUtil` (not shown). The methods `f1()` and `f2()` implement the forward and inverse gamma correction of CIELAB components (see Eqns. (14.6) and (14.13)).

```
  // sRGB→CIELab conversion:
  public float[] fromRGB(float[] srgb) {
    // get linear rgb components:
    double r = sRgbUtil.gammaInv(srgb[0]);
    double g = sRgbUtil.gammaInv(srgb[1]);
    double b = sRgbUtil.gammaInv(srgb[2]);
    // convert to XYZ (D65-based, Eqn. (14.29)):
    float X =
      (float) (0.412453 * r + 0.357580 * g + 0.180423 * b);
    float Y =
      (float) (0.212671 * r + 0.715160 * g + 0.072169 * b);
    float Z =
      (float) (0.019334 * r + 0.119193 * g + 0.950227 * b);
    float[] XYZ65 = new float[] {X, Y, Z};
    return fromCIEXYZ65(XYZ65);
  }

  // CIELab→sRGB conversion:
  public float[] toRGB(float[] Lab) {
    float[] XYZ65 = toCIEXYZ65(Lab);
    double X = XYZ65[0];
    double Y = XYZ65[1];
    double Z = XYZ65[2];
    // XYZ→RGB (linear components, Eqn. (14.28)):
    double r = ( 3.240479*X + -1.537150*Y + -0.498535*Z);
    double g = (-0.969256*X +  1.875992*Y +  0.041556*Z);
    double b = ( 0.055648*X + -0.204043*Y +  1.057311*Z);
    // RGB→sRGB (nonlinear components):
    float rr = (float) sRgbUtil.gammaFwd(r);
    float gg = (float) sRgbUtil.gammaFwd(g);
    float bb = (float) sRgbUtil.gammaFwd(b);
    return new float[] {rr, gg, bb};
  }

  static final double epsilon = 216.0 / 24389;   // Eqn. (14.9)
  static final double kappa = 841.0 / 108;   // Eqn. (14.10)

  // Gamma correction for L* (forward, Eqn. (14.8)):
  double f1 (double c) {
    if (c > epsilon) // 0.008856
      return Math.cbrt(c);
    elses
      return (kappa * c) + (16.0 / 116);
  }

  // Gamma correction for L* (inverse, Eqn. (14.15)):
  double f2 (double c) {
    double c3 = c * c * c;
    if (c3 > epsilon)
      return c3;
    else
      return (c - 16.0 / 116) / kappa;
  }

} // end of class LabColorSpace
```

```
// convert to sRGB:
float[] RGBColor = scannerCs.toRGB(deviceColor);

// convert to (D50-based) XYZ:
float[] XYZColor = scannerCs.toCIEXYZ(deviceColor);
```

Similarly, we can calculate the accurate color values to be sent to the monitor by creating a suitable color space object from this device's ICC profile.

14.8 Exercises

Exercise 14.1. For chromatic adaptation (defined in Eqn. (14.43)), transformation matrices other than the Bradford model (Eqn. (14.45)) have been proposed; for example, [225],

$$\boldsymbol{M}_{\mathrm{CAT}}^{(2)} = \begin{pmatrix} 1.2694 & -0.0988 & -0.1706 \\ -0.8364 & 1.8006 & 0.0357 \\ 0.0297 & -0.0315 & 1.0018 \end{pmatrix} \quad \text{or} \tag{14.50}$$

$$\boldsymbol{M}_{\mathrm{CAT}}^{(3)} = \begin{pmatrix} 0.7982 & 0.3389 & -0.1371 \\ -0.5918 & 1.5512 & 0.0406 \\ 0.0008 & -0.0239 & 0.9753 \end{pmatrix}. \tag{14.51}$$

Derive the complete chromatic adaptation transformations $\boldsymbol{M}_{50|65}$ and $\boldsymbol{M}_{65|50}$ for converting between D65 and D50 colors, analogous to Eqns. (14.46) and (14.47), for each of the above transformation matrices.

Exercise 14.2. Implement the conversion of an sRGB color image to a colorless (grayscale) sRGB image using the three methods in Eqn. (14.37) (incorrectly applying standard weights to nonlinear $R'G'B'$ components), Eqn. (14.38) (exact computation), and Eqn. (14.39) (approximation using nonlinear components and modified weights). Compare the results by computing difference images, and also determine the total errors.

Exercise 14.3. Write a program to evaluate the errors that are introduced by using *nonlinear* instead of linear color components for grayscale conversion. To do this, compute the diffence between the Y values obtained with the linear variant (Eqn. (14.38)) and the nonlinear variant (Eqn. (14.39) with $w'_R = 0.309$, $w'_G = 0.609$, $w'_B = 0.082$) for all possible 2^{24} RGB colors. Let your program return the maximum gray value difference and the sum of the absolute differences for all colors.

Exercise 14.4. Determine the virtual primaries $\mathbf{R}^*, \mathbf{G}^*, \mathbf{B}^*$ obtained by Bradford adaptation (Eqn. (14.42)), with $\boldsymbol{M}_{\mathrm{CAT}}$ as defined in Eqn. (14.45). What are the resulting coordinates in the xy chromaticity diagram? Are the primaries inside the visible color range?