Frequently Asked Questions about Color

Charles Poynton

www.inforamp.net/~poynton
poynton@poynton.com

This FAQ is intended to clarify aspects of color that are important to color image coding, computer graphics, image processing, video, and the transfer of digital images to print.

I assume that you are familiar with intensity, luminance (CIE *Y*), lightness (CIE *L**), and the nonlinear relationship between CRT voltage and luminance – that is, gamma. To learn more about these topics, please read the companion *Frequently Asked Questions about Gamma*.

This document is available on the Internet from Toronto at: <http://www.inforamp.net/~poynton/>

Table of Contents

1 What is color?

Color is the perceptual result of light in the visible region of the spectrum, having wavelengths in the region of 400 nm to 700 nm, incident upon the retina. Physical power (or *radiance*) is expressed in a *spectral power distribution* (SPD), often in 31 components each representing a 10 nm band.

The human retina has three types of color photoreceptor *cone* cells, which respond to incident radiation with somewhat different spectral response curves. A fourth type of photoreceptor cell, the *rod*, is also present in the retina. Rods are effective only at extremely low light levels (colloquially, *night vision*), and although important for vision play no role in image reproduction.

Because there are exactly three types of color photoreceptor, three numerical components are necessary and sufficient to describe a color, providing that appropriate spectral weighting functions are used. This is the concern of the science of *colorimetry*. In 1931, the Commission Internationale de L'Éclairage (CIE) adopted standard curves for a hypothetical *Standard Observer*. These curves specify how an SPD can be transformed into a set of three numbers that specifies a color.

The CIE system is immediately and almost universally applicable to self-luminous sources and displays. However the colors produced by reflective systems such as photography, printing or paint are a function not only of the colorants but also of the SPD of the ambient illumination. If your application has a strong dependence upon the spectrum of the illuminant, you may have to resort to spectral matching.

Sir Isaac Newton said, "Indeed rays, properly expressed, are not colored." SPDs exist in the physical world, but colour exists only in the eye and the brain.

Berlin and Kay [1] state that although different languages encode in their vocabularies different numbers of basic color categories, a total universal inventory of exactly eleven basic color categories exists from which the eleven or fewer basic color terms of any given language are always drawn. The eleven basic color categories are white, black, red, green, yellow, blue, brown, purple, pink, orange, and gray.

2 What are *brightness, intensity, luminance*, and *lightness*?

Pixel values in accurate grayscale images are based upon *luminance*. Pixel values in accurate color images are based upon *tristimulus values*. To understand luminance and tristimulus values, you must be familiar with these terms: *radiant intensity, radiance, luminous intensity, luminance*, and *lightness*. For explanations of these important terms, see the companion *Gamma FAQ*.

3 What are *tristimulus values*?

Color images are sensed and reproduced based upon *tristimulus values*, whose amplitude is proportional to radiance, but whose spectral composition is carefully chosen according to the principles of color science. (I call these values linear-light.) As their name implies, tristimulus values come in sets of 3. In most imaging systems, *RGB* tristimulus values are subjected to a nonlinear transfer function – gamma correction – that mimics the lightness response of vision. The notation *R'G'B'* denotes the nonlinearity. Most imaging systems use *RGB* values that are not proportional to tristimulus values.

4 What is *hue*?

According to the CIE [2], *hue is the attribute of a visual sensation according to which an area appears to be similar to one of the perceived colours, red, yellow, green and blue, or a combination of two of them*. Roughly speaking, if the dominant wavelength of an SPD shifts, the hue of the associated color will shift.

5 What is *saturation*?

Again from the CIE, *saturation is the colourfulness of an area judged in proportion to its brightness*. Saturation runs from neutral gray through pastel to saturated colors. Roughly speaking, the more an SPD is concentrated at one wavelength, the more saturated will be the associated color. You can desaturate a color by adding light that contains power at all wavelengths.

6 How is color specified?

The CIE system defines how to map an SPD to a *triple* of numerical components that are the mathematical coordinates of color space. Think of these as coordinates of three-dimensional color space. Their function is analogous to coordinates on a map. Cartographers have different map projections for different functions: some map projections preserve areas, others show latitudes and longitudes as straight lines. No single map projection fills all the needs of map users. Similarly, no single color system fills all of the needs of color users.

The systems useful today for color specification include CIE *XYZ*, CIE *xyY*, CIE *L*u*v** and CIE *L*a*b**. Numerical values of hue and saturation are not very useful for color specification, for reasons to be discussed in section 35.

A color specification system needs to be able to represent any color with high precision. Since few colors are handled at a time, a specification system can be computationally complex. Any system for color specification must be intimately related to the CIE specifications.

You can specify a single "spot" color using a *color order system* such as Munsell. Systems like Munsell come with swatch books to enable visual color matches, and have documented methods of transforming between coordinates in the system and CIE values. Systems like Munsell are not useful for image data. You can specify an ink color by specifying the proportions of standard (or secret) inks that can be mixed to make the color. That's how PANTONE™ works. Although widespread, it's proprietary. No translation to CIE is publicly available.

7 Should I use a color specification system for image data?

A digitized color image is represented as an array of pixels, where each pixel contains numerical components that define a color. Three components are necessary and sufficient for this purpose, although in printing it is convenient to use a fourth (black) component.

In theory, the three numerical values for image coding could be provided by a color specification system. But a practical image coding system needs to be computationally efficient, cannot afford unlimited precision, need not be intimately related to the CIE system and generally needs to cover only a reasonably wide range of colors and not all of the colors. So image coding uses different systems than color specification.

The systems useful for image coding are linear RGB, nonlinear *R'G'B'*, nonlinear CMY, nonlinear CMYK, and derivatives of nonlinear *R'G'B'*

such as $Y'C_BC_R$. Numerical values of hue and saturation are not useful in color image coding.

If you manufacture cars, you have to match the color of paint on the door with the color of paint on the fender. A color specification system will be necessary. But to convey a picture of the car, you need image coding. You can afford to do quite a bit of computation in the first case because you have only two colored elements, the door and the fender. In the second case, the color coding must be quite efficient because you may have a million colored elements or more.

For a highly readable short introduction to color image coding, see DeMarsh and Giorgianni [3]. For a terse, complete technical treatment, read Schreiber [4].

8 What weighting of red, green and blue corresponds to luminance?

Direct acquisition of luminance requires use of a very specific spectral weighting. However, luminance can also be computed as a weighted sum of red, green and blue components.

If three sources appear red, green and blue, and have the same radiance in the visible spectrum, then the green will appear the brightest of the three because the luminous efficiency function peaks in the green region of the spectrum. The red will appear less bright, and the blue will be the darkest of the three. As a consequence of the luminous efficiency function, all saturated blue colors are quite dark and all saturated yellows are quite light. If luminance is computed from red, green and blue, the coefficients will be a function of the particular red, green and blue spectral weighting functions employed, but the green coefficient will be quite large, the red will have an intermediate value, and the blue coefficient will be the smallest of the three.

Contemporary CRT phosphors are standardized in Rec. 709 [9], to be described in section 16. The weights to compute true CIE luminance from linear red, green and blue (indicated without prime symbols), for the Rec. 709, are these:

$$Y_{709} = 0.2126\,R + 0.7152\,G + 0.0722\,B$$

This computation assumes that the luminance spectral weighting can be formed as a linear combination of the scanner curves, and assumes that the component signals represent linear-light. Either or both of these conditions can be relaxed to some extent depending on the application.

Some computer systems have computed brightness using $(R+G+B)/3$. This is at odds with the properties of human vision, as will be discussed under *What are HSB and HLS?* in section 35.

The coefficients 0.299, 0.587 and 0.114 properly computed luminance for monitors having phosphors that were contemporary at the introduction of NTSC television in 1953. They are still appropriate for computing video *luma* to be discussed below in section 10. However, these coefficients do not accurately compute luminance for contemporary monitors.

9 Can blue be assigned fewer bits than red or green?

Blue has a small contribution to the brightness sensation. However, human vision has extraordinarily good color discrimination capability in

blue colors. So if you give blue fewer bits than red or green, you will introduce noticeable contouring in blue areas of your pictures.

10 What is *luma*?

It is useful in a video system to convey a component representative of luminance and two other components representative of color. It is important to convey the component representative of luminance in such a way that noise (or quantization) introduced in transmission, processing and storage has a perceptually similar effect across the entire tone scale from black to white. The ideal way to accomplish these goals would be to form a luminance signal by matrixing *RGB*, then subjecting luminance to a nonlinear transfer function similar to the *L** function.

There are practical reasons in video to perform these operations in the opposite order. First a nonlinear transfer function – *gamma correction* – is applied to each of the linear *R*, *G* and *B*. Then a weighted sum of the nonlinear components is computed to form a signal representative of luminance. The resulting component is related to brightness but is not CIE luminance. Many video engineers call it *luma* and give it the symbol *Y'*. It is often carelessly called *luminance* and given the symbol *Y*. You must be careful to determine whether a particular author assigns a linear or nonlinear interpretation to the term *luminance* and the symbol *Y*.

The coefficients that correspond to the "NTSC" red, green and blue CRT phosphors of 1953 are standardized in ITU-R Recommendation BT. 601 (formerly CCIR Rec. 601). I call it *Rec. 601*. To compute nonlinear video *luma* from nonlinear red, green and blue:

$$Y'_{601} = 0.299R' + 0.587G' + 0.114B'$$

The prime symbols in this equation, and in those to follow, denote nonlinear components.

11 What are CIE *XYZ* components?

The CIE system is based on the description of color as a luminance component Y, as described above, and two additional components X and Z. The spectral weighting curves of X and Z have been standardized by the CIE based on statistics from experiments involving human observers. XYZ *tristimulus values* can describe any color. (*RGB* tristimulus values will be described later.)

The magnitudes of the *XYZ* components are proportional to physical energy, but their spectral composition corresponds to the color matching characteristics of human vision.

The CIE system is defined in Publication CIE No 15.2, *Colorimetry, Second Edition* (1986) [5].

12 Does my scanner use the CIE spectral curves?

Probably not. Scanners are most often used to scan images such as color photographs and color offset prints that are already "records" of three components of color information. The usual task of a scanner is not spectral analysis but extraction of the values of the three components that have already been recorded. Narrowband filters are more suited to this task than filters that adhere to the principles of colorimetry.

If you place on your scanner an original colored object that has "original" SPDs that are not already a record of three components, chances are your

scanner will not very report accurate *RGB* values. This is because most scanners do not conform very closely to CIE standards.

13 What are CIE *x* and *y* chromaticity coordinates?

It is often convenient to discuss "pure" color in the absence of brightness. The CIE defines a normalization process to compute "little" *x* and *y chromaticity coordinates:*

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

A color plots as a point in an (*x, y*) *chromaticity diagram*. When a narrowband SPD comprising power at just one wavelength is swept across the range 400 nm to 700 nm, it traces a shark-fin shaped *spectral locus* in (*x, y*) coordinates. The sensation of purple cannot be produced by a single wavelength: to produce purple requires a mixture of shortwave and longwave light. The *line of purples* on a chromaticity diagram joins extreme blue to extreme red. All colors are contained in the area in (*x, y*) bounded by the line of purples and the spectral locus.

A color can be specified by its chromaticity and luminance, in the form of an *xyY* triple. To recover *X* and *Z* from chromaticities and luminance, use these relations:

$$X = \frac{x}{y}Y \qquad Z = \frac{1-x-y}{y}Y$$

The bible of color science is Wyszecki and Styles, *Color Science* [6]. But it's daunting. For Wyszecki's own condensed version, see *Color in Business, Science and Industry, Third Edition* [7]. It is directed to the color industry: ink, paint and the like. For an approachable introduction to the same theory, accompanied by descriptions of image reproduction, see R.W.G. Hunt, *The Reproduction of Colour, Fifth Edition* [8].

14 What is white?

In additive image reproduction, the *white point* is the chromaticity of the color reproduced by equal red, green and blue components. White point is a function of the ratio (or *balance*) of power among the primaries. In subtractive reproduction, white is the SPD of the illumination, multiplied by the SPD of the media. There is no unique physical or perceptual definition of white, so to achieve accurate color interchange you must specify the characteristics of your white.

It is often convenient for purposes of calculation to define white as a uniform SPD. This white reference is known as the *equal-energy illuminant*, or *CIE Illuminant E*.

A more realistic reference that approximates daylight has been specified numerically by the CIE as Illuminant D_{65}. You should use this unless you have a good reason to use something else. The print industry commonly uses D_{50} and photography commonly uses D_{55}. These represent compromises between the conditions of indoor (tungsten) and daylight viewing.

15 What is color temperature?

Planck determined that the SPD radiated from a hot object – a *black body radiator* – is a function of the temperature to which the object is heated. Many sources of illumination have, at their core, a heated object, so it is often useful to characterize an illuminant by specifying the temperature

(in units of kelvin, K) of a black body radiator that appears to have the same hue.

Although an illuminant can be specified informally by its color temperature, a more complete specification is provided by the chromaticity coordinates of the SPD of the source.

Modern blue CRT phosphors are more efficient with respect to human vision than red or green. In a quest for brightness at the expense of color accuracy, it is common for a computer display to have excessive blue content, about twice as blue as daylight, with white at about 9300 K.

Human vision adapts to white in the viewing environment. An image viewed in isolation – such as a slide projected in a dark room – creates its own white reference, and a viewer will be quite tolerant of errors in the white point. But if the same image is viewed in the presence of an external white reference or a second image, then differences in white point can be objectionable.

Complete adaptation seems to be confined to the range 5000 K to 5500 K. For most people, D_{65} has a little hint of blue. Tungsten illumination, at about 3200 K, always appears somewhat yellow.

16 How can I characterize red, green, and blue?

Additive reproduction is based on physical devices that produce all-positive SPDs for each primary. Physically and mathematically, the spectra add. The largest range of colors will be produced with primaries that appear red, green and blue. Human color vision obeys the principle of superposition, so the color produced by any additive mixture of three primary spectra can be predicted by adding the corresponding fractions of the *XYZ* components of the primaries: the colors that can be mixed from a particular set of RGB primaries are completely determined by the colors of the primaries by themselves. Subtractive reproduction is much more complicated: the colors of mixtures are determined by the primaries and by the colors of their combinations.

An additive RGB system is specified by the chromaticities of its primaries and its white point. The extent (*gamut*) of the colors that can be mixed from a given set of RGB primaries is given in the (x, y) chromaticity diagram by a triangle whose vertices are the chromaticities of the primaries.

In computing there are no standard primaries or white point. If you have an RGB image but have no information about its chromaticities, you cannot accurately reproduce the image.

The NTSC in 1953 specified a set of primaries that were representative of phosphors used in color CRTs of that era. But phosphors changed over the years, primarily in response to market pressures for brighter receivers, and by the time of the first the videotape recorder the primaries in use were quite different than those "on the books". So although you may see the NTSC primary chromaticities documented, they are of no use today.

Contemporary studio monitors have slightly different standards in North America, Europe and Japan. But international agreement has been obtained on primaries for high definition television (HDTV), and these

primaries are closely representative of contemporary monitors in studio video, computing and computer graphics. The primaries and the D_{65} white point of *Rec. 709* [9] are:

	R	G	B	white
x	0.640	0.300	0.150	0.3127
y	0.330	0.600	0.060	0.3290
z	0.030	0.100	0.790	0.3582

For a discussion of nonlinear RGB in computer graphics, see Lindbloom [10]. For technical details on monitor calibration, consult Cowan [11].

17 How do I transform between CIE *XYZ* and a particular set of *RGB* primaries?

RGB values in a particular set of primaries can be transformed to and from CIE *XYZ* by a three-by-three matrix transform. These transforms involve *tristimulus values*, that is, sets of three linear-light components that conform to the CIE color matching functions. CIE *XYZ* is a special case of tristimulus values. In *XYZ*, any color is represented by a positive set of values.

Details can be found in SMPTE RP 177-1993 [12].

To transform from CIE XYZ into Rec. 709 *RGB* (with its D_{65} white point), use this transform:

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

This matrix has some negative coefficients: XYZ colors that are *out of gamut* for a particular *RGB* transform to *RGB* where one or more *RGB* components is negative or greater than unity.

Here's the inverse transform. Because white is normalized to unity, the middle row sums to unity:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \bullet \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix}$$

To recover primary chromaticities from such a matrix, compute little *x* and *y* for each *RGB* column vector. To recover the white point, transform RGB=[1, 1, 1] to XYZ, then compute *x* and *y*.

18 Is *RGB* always device-dependent?

Video standards specify abstract *R'G'B'* systems that are closely matched to the characteristics of real monitors. Physical devices that produce additive color involve tolerances and uncertainties, but if you have a monitor that conforms to Rec. 709 within some tolerance, you can consider the monitor to be device-independent.

The importance of Rec. 709 as an interchange standard in studio video, broadcast television and high definition television, and the perceptual basis of the standard, assures that its parameters will be used even by devices such as flat-panel displays that do not have the same physics as CRTs.

19 How do I transform data from one set of *RGB* primaries to another?

RGB values in a system employing one set of primaries can be transformed into another set by a three-by-three linear-light matrix transform. Generally these matrices are normalized for a white point luminance of unity. For details, see *Television Engineering Handbook* [13].

As an example, here is the transform from SMPTE 240M (or SMPTE RP 145) *RGB* to Rec. 709:

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 0.939555 & 0.050173 & 0.010272 \\ 0.017775 & 0.965795 & 0.016430 \\ -0.001622 & -0.004371 & 1.005993 \end{bmatrix} \bullet \begin{bmatrix} R_{240M} \\ G_{240M} \\ B_{240M} \end{bmatrix}$$

All of these terms are close to either zero or one. In a case like this, if the transform is computed in the nonlinear (gamma-corrected) *R′G′B′* domain the resulting errors will be insignificant.

Here's another example. To transform EBU 3213 RGB to Rec. 709:

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 1.044036 & -0.044036 & 0. \\ 0. & 1. & 0. \\ 0. & 0.011797 & 0.988203 \end{bmatrix} \bullet \begin{bmatrix} R_{EBU} \\ G_{EBU} \\ B_{EBU} \end{bmatrix}$$

Transforming among *RGB* systems may lead to an *out of gamut RGB* result where one or more *RGB* components is negative or greater than unity.

20 Should I use *RGB* or *XYZ* for image synthesis?

Once light is on its way to the eye, any tristimulus-based system will work. But the interaction of light and objects involves spectra, not tristimulus values. In synthetic computer graphics, the calculations are actually simulating sampled SPDs, even if only three components are used. Details concerning the resultant errors are found in Hall [14].

21 What is subtractive color?

Subtractive systems involve colored dyes or filters that absorb power from selected regions of the spectrum. The three filters are placed in tandem. A dye that appears cyan absorbs longwave (red) light. By controlling the amount of cyan dye (or ink), you modulate the amount of red in the image.

In physical terms the spectral transmission curves of the colorants multiply, so this method of color reproduction should really be called "multiplicative". Photographers and printers have for decades measured transmission in base-10 logarithmic *density* units, where transmission of unity corresponds to a density of 0, transmission of 0.1 corresponds to a density of 1, transmission of 0.01 corresponds to a density of 2 and so on. When a printer or photographer computes the effect of filters in tandem, he subtracts density values instead of multiplying transmission values, so he calls the system *subtractive*.

To achieve a wide range of colors in a subtractive system requires filters that appear colored cyan, yellow and magenta (*CMY*). Cyan in tandem with magenta produces blue, cyan with yellow produces green, and

magenta with yellow produces red. Smadar Nehab suggests this memory aid:

Additive primaries are at the top, subtractive at the bottom. On the left, magenta and yellow filters combine to produce red. On the right, red and green sources add to produce yellow.

22 Why did my grade three teacher tell me that the primaries are red, yellow and blue?

To get a wide range of colors in an additive system, the primaries must appear red, green and blue (*RGB*). In a subtractive system the primaries must appear yellow, cyan and magenta (*CMY*). It is complicated to predict the colors produced when mixing paints, but roughly speaking, paints mix additively to the extent that they are opaque (like oil paints), and subtractively to the extent that they are transparent (like watercolors). This question also relates to color names: your grade three "red" was probably a little on the magenta side, and "blue" was probably quite cyan. For a discussion of paint mixing from a computer graphics perspective, consult Haase [15].

23 Is *CMY* just one-minus-*RGB*?

In a theoretical subtractive system, *CMY* filters could have spectral absorption curves with no overlap. The color reproduction of the system would correspond exactly to additive color reproduction using the red, green and blue primaries that resulted from pairs of filters in combination.

Practical photographic dyes and offset printing inks have spectral absorption curves that overlap significantly. Most magenta dyes absorb mediumwave (green) light as expected, but incidentally absorb about half that amount of shortwave (blue) light. If reproduction of a color, say brown, requires absorption of all shortwave light then the incidental absorption from the magenta dye is not noticed. But for other colors, the so-called one-minus-*RGB* formula produces mixtures with much less blue than expected, and therefore produce pictures that have a yellow cast in the mid tones. Similar but less severe interactions are evident for the other pairs of practical inks and dyes.

Due to the spectral overlap among the colorants, converting *CMY* using the one-minus-*RGB* method works for applications such as business graphics where accurate color need not be preserved, but the method fails to produce acceptable color images.

Multiplicative mixture in a *CMY* system is mathematically nonlinear, and the effect of the unwanted absorptions cannot be easily analyzed or compensated. The colors that can be mixed from a particular set of *CMY* primaries cannot be determined from the colors of the primaries themselves, but are also a function of the colors of the sets of combinations of the primaries.

Print and photographic reproduction is also complicated by nonlinearities in the response of the three (or four) channels. In offset printing, the physical and optical processes of *dot gain* introduce nonlinearity that is roughly comparable to gamma correction in video. In a typical system used for print, a black code of 128 (on a scale of 0 to 255) produces a

reflectance of about 0.26, not the 0.5 that you would expect from a linear system. Computations cannot be meaningfully performed on *CMY* components without taking nonlinearity into account.

For a detailed discussion of transferring colorimetric image data to print media, see Stone [16].

24 Why does offset printing use black ink in addition to *CMY*?

Printing black by overlaying cyan, yellow and magenta ink in offset printing has three major problems. First, colored ink is expensive. Replacing colored ink by black ink – which is primarily carbon – makes economic sense. Second, printing three ink layers causes the printed paper to become quite wet. If three inks can be replaced by one, the ink will dry more quickly, the press can be run faster, and the job will be less expensive. Third, if black is printed by combining three inks, and mechanical tolerances cause the three inks to be printed slightly out of register, then black edges will suffer colored tinges. Vision is most demanding of spatial detail in black and white areas. Printing black with a single ink minimizes the visibility of registration errors.

Other printing processes may or may not be subject to similar constraints.

25 What are color differences?

This term is ambiguous. In its first sense, *color difference* refers to numerical differences between color specifications. The perception of color differences in *XYZ* or *RGB* is highly nonuniform. The study of *perceptual uniformity* concerns numerical differences that correspond to color differences at the threshold of perceptibility (*just noticeable differences*, or JNDs).

In its second sense, *color difference* refers to color components where brightness is "removed". Vision has poor response to spatial detail in colored areas of the same luminance, compared to its response to luminance spatial detail. If data capacity is at a premium it is advantageous to transmit luminance with full detail and to form two *color difference* components each having no contribution from luminance. The two color components can then have spatial detail removed by filtering, and can be transmitted with substantially less information capacity than luminance.

Instead of using a true luminance component to represent brightness, it is ubiquitous for practical reasons to use a *luma* signal that is computed nonlinearly as outlined above (*What is luma?*).

The easiest way to "remove" brightness information to form two color channels is to subtract it. The luma component already contains a large fraction of the green information from the image, so it is standard to form the other two components by subtracting luma from nonlinear blue (to form B'-Y') and by subtracting luma from nonlinear red (to form R'-Y'). These are called *chroma*.

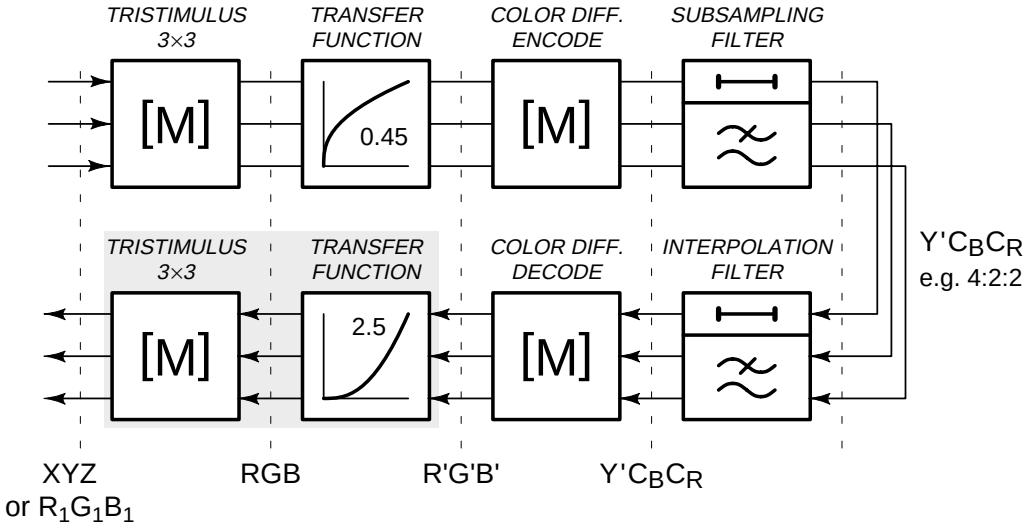
Various scale factors are applied to (B'-Y') and (R'-Y') for different applications. The $Y'P_BP_R$ scale factors are optimized for component analog video. The $Y'C_BC_R$ scaling is appropriate for component digital video such as studio video, JPEG and MPEG. Kodak's PhotoYCC™ uses scale factors optimized for the gamut of film colors. $Y'UV$ scaling is appropriate as an intermediate step in the formation of composite NTSC or PAL video signals, but is not appropriate when the components are kept sepa-

rate. The *Y′UV* nomenclature is now used rather loosely, and it sometimes denotes any scaling of (*B′-Y′*) and (*R′-Y′*). *Y′IQ* coding is obsolete.

The subscripts in C_BC_R and P_BP_R are often written in lower case. I find this to compromise readability, so without introducing any ambiguity I write them in uppercase. Authors with great attention to detail sometimes "prime" these quantities to indicate their nonlinear nature, but because no practical image coding system employs linear color differences I consider it safe to omit the primes.

26 How do I obtain color difference components from tristimulus values?

Here is the block diagram for luma/color difference encoding and decoding:



From linear *XYZ* – or linear $R_1G_1B_1$ whose chromaticity coordinates are different from the interchange standard – apply a 3×3 matrix transform to obtain linear *RGB* according to the interchange primaries. Apply a nonlinear transfer function ("gamma correction") to each of the components to get nonlinear *R′G′B′*. Apply a 3×3 matrix to obtain color difference components such as $Y'P_BP_R$, $Y'C_BC_R$ or PhotoYCC. If necessary, apply a color subsampling filter to obtain subsampled color difference components. To decode, invert the above procedure: run through the block diagram right-to-left using the inverse operations. If your monitor conforms to the interchange primaries, decoding need not explicitly use a transfer function or the tristimulus 3×3.

The block diagram emphasizes that 3×3 matrix transforms are used for two distinctly different tasks. When someone hands you a 3×3, you have to ask for which task it is intended.

27 How do I encode $Y'P_BP_R$ components?

Although the following matrices could in theory be used for tristimulus signals, it is ubiquitous to use them with gamma-corrected signals.

To encode $Y'P_BP_R$, start with the basic *Y′*, (*B′-Y′*) and (*R′-Y′*) relationships:

Eq 1

$$\begin{bmatrix} Y'_{601} \\ B'-Y'_{601} \\ R'-Y'_{601} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \bullet \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

$Y'P_BP_R$ components have unity excursion, where Y' ranges [0..+1] and each of P_B and P_R ranges [-0.5..+0.5]. The (B'-Y') and (R'-Y') rows need to be scaled by $\frac{0.5}{0.886}$ and $\frac{0.5}{0.701}$. To encode from $R'G'B'$ where reference black is 0 and reference white is +1:

Eq 2

$$\begin{bmatrix} Y'_{601} \\ P_B \\ P_R \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \bullet \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

The first row comprises the luma coefficients; these sum to unity. The second and third rows each sum to zero, a necessity for color difference components. The +0.5 entries reflect the maximum excursion of P_B and P_R of +0.5, for the blue and red primaries [0, 0, 1] and [1, 0, 0].

The inverse, decoding matrix is this:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344136 & -0.714136 \\ 1 & 1.772 & 0 \end{bmatrix} \bullet \begin{bmatrix} Y'_{601} \\ P_B \\ P_R \end{bmatrix}$$

28 How do I encode $Y'C_BC_R$ components from $R'G'B'$ in [0, +1]?

Rec. 601 specifies eight-bit coding where Y' has an excursion of 219 and an offset of +16. This coding places black at code 16 and white at code 235, reserving the extremes of the range for signal processing headroom and footroom. C_B and C_R have excursions of ±112 and offset of +128, for a range of 16 through 240 inclusive.

To compute $Y'C_BC_R$ from $R'G'B'$ in the range [0..+1], scale the rows of the matrix of Eq 2 by the factors 219, 224 and 224, corresponding to the excursions of each of the components:

Eq 3

$$\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112. \\ 112. & -93.786 & -18.214 \end{bmatrix} \bullet \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Summing the first row of the matrix yields 219, the luma excursion from black to white. The two entries of 112 reflect the positive C_BC_R extrema of the blue and red primaries.

Clamp all three components to the range 1 through 254 inclusive, since Rec. 601 reserves codes 0 and 255 for synchronization signals.

To recover $R'G'B'$ in the range [0..+1] from $Y'C_BC_R$, use the inverse of Eq 3 above:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 0.00456621 & 0. & 0.00625893 \\ 0.00456621 & -0.00153632 & -0.00318811 \\ 0.00456621 & 0.00791071 & 0. \end{bmatrix} \bullet \left(\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

This looks overwhelming, but the $Y'C_BC_R$ components are integers in eight bits and the reconstructed $R'G'B'$ are scaled down to the range [0..+1].

29 How do I encode $Y'C_BC_R$ components from computer $R'G'B'$?

In computing it is conventional to use eight-bit coding with black at code 0 and white at 255. To encode $Y'C_BC_R$ from $R'G'B'$ in the range [0..255], using eight-bit binary arithmetic, scale the $Y'C_BC_R$ matrix of Eq 3 by $^{256}/_{255}$:

$$\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \bullet \begin{bmatrix} R'_{255} \\ G'_{255} \\ B'_{255} \end{bmatrix}$$

To decode $R'G'B'$ in the range [0..255] from Rec. 601 $Y'C_BC_R$, using eight-bit binary arithmetic:

Eq 4

$$\begin{bmatrix} R'_{255} \\ G'_{255} \\ B'_{255} \end{bmatrix} = \frac{1}{256} \begin{bmatrix} 298.082 & 0. & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0. \end{bmatrix} \bullet \left(\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

The multiplications by $^1/_{256}$ can be accomplished by shifting. Some of the coefficients, when scaled by $^1/_{256}$, are larger than unity. These coefficients will need more than eight multiplier bits.

For implementation in binary arithmetic the matrix coefficients have to be rounded. When you round, take care to preserve the row sums of [1, 0, 0].

The matrix of Eq 4 will decode standard $Y'C_BC_R$ components to *RGB* components in the range [0..255], subject to roundoff error. You must take care to avoid overflow due to roundoff error. But you must protect against overflow in any case, because studio video signals use the extremes of the coding range to handle signal overshoot and undershoot, and these will require clipping when decoded to an RGB range that has no headroom or footroom.

30 How do I encode $Y'C_BC_R$ components from studio video?

Studio $R'G'B'$ signals use the same 219 excursion as the luma component of $Y'C_BC_R$.To encode $Y'C_BC_R$ from $R'G'B'$ in the range [0..219], using eight-bit binary arithmetic, scale the $Y'C_BC_R$ encoding matrix of Eq 3 above by $^{256}/_{219}$. Here is the encoding transform for studio video:

$$\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 76.544 & 150.272 & 29.184 \\ -44.182 & -86.740 & 130.922 \\ 130.922 & -109.631 & -21.291 \end{bmatrix} \bullet \begin{bmatrix} R'_{219} \\ G'_{219} \\ B'_{219} \end{bmatrix}$$

To decode $R'G'B'$ in the range [0..219] from $Y'C_BC_R$, using eight-bit binary arithmetic:

$$\begin{bmatrix} R'_{219} \\ G'_{219} \\ B'_{219} \end{bmatrix} = \frac{1}{256} \begin{bmatrix} 256. & 0. & 350.901 \\ 256. & -86.132 & -178.738 \\ 256. & 443.506 & 0. \end{bmatrix} \bullet \left(\begin{bmatrix} Y'_{601} \\ C_B \\ C_R \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right)$$

The entries of 256 in this matrix indicate that the corresponding component can simply be added; there is no need for a multiplication operation. This matrix contains entries larger than 256; the corresponding multipliers will need capability for nine bits.

The matrices in this section conform to Rec. 601 and apply directly to conventional 525/59.94 and 625/50 video. It is not yet decided whether

emerging HDTV standards will use the same matrices, or adopt a new set of matrices having different luma coefficients. In my view it would be unfortunate if different matrices were adopted, because then image coding and decoding would depend on whether the picture was small (conventional video) or large (HDTV).

In digital video, Rec. 601 standardizes subsampling denoted 4:2:2, where C_B and C_R components are subsampled horizontally by a factor of two with respect to luma. JPEG and MPEG conventionally subsample by a factor of two in the vertical dimension as well, denoted 4:2:0.

Color difference coding is standardized in Rec. 601. For details on color difference coding as used in video, consult Poynton [17].

31 How do I decode *R′G′B′* from PhotoYCC™?

Kodak's PhotoYCC uses the Rec. 709 primaries, white point and transfer function. Reference white codes to luma 189; this preserves film highlights. The color difference coding is asymmetrical, to encompass film gamut. You are unlikely to encounter any raw image data in PhotoYCC form because YCC is closely associated with the PhotoCD™ system whose compression methods are proprietary. But just in case, the following equation is comparable to in that it produces *R′G′B′* in the range [0..+1] from integer YCC. If you want to return *R′G′B′* in a different range, or implement the equation in eight-bit integer arithmetic, use the techniques in the section above.

$$\begin{bmatrix} R'_{709} \\ G'_{709} \\ B'_{709} \end{bmatrix} = \begin{bmatrix} 0.0054980 & 0. & 0.0051681 \\ 0.0054980 & -0.0015446 & -0.0026325 \\ 0.0054980 & 0.0079533 & 0. \end{bmatrix} \bullet \left(\begin{bmatrix} Y'_{601,189} \\ C_1 \\ C_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 156 \\ 137 \end{bmatrix} \right)$$

Decoded *R′G′B′* components from PhotoYCC can exceed unity or go below zero. PhotoYCC extends the Rec. 709 transfer function above unity, and reflects it around zero, to accommodate wide excursions of *R′G′B′*. To decode to CRT primaries, clip *R′G′B′* to the range zero to one.

32 Will you tell me how to decode *Y′UV* and *Y′IQ*?

No, I won't! *Y′UV* and *Y′IQ* have scale factors appropriate to composite NTSC and PAL. They have no place in component digital video! You shouldn't code into these systems, and if someone hands you an image claiming it's *Y′UV*, chances are it's actually $Y'C_BC_R$, it's got the wrong scale factors, or it's linear-light.

Well OK, just this once. To transform *Y′*, (*B′*-*Y′*) and (*R′*-*Y′*) components from Eq 1 to *Y′UV*, scale (*B′*-*Y′*) by 0.492111 to get U and scale *R′*-*Y′* by 0.877283 to get V. The factors are chosen to limit composite NTSC or PAL amplitude for all legal *R′G′B′* values:

$$-\frac{1}{3} \le \left[Y' \pm \sqrt{0.492111\left(B' - Y'\right) + 0.877283\left(R' - Y'\right)} \right] \le \frac{4}{3}$$

To transform from *Y′IQ* to *Y′UV*, perform a 33° rotation and an exchange of color difference axes:

$$\begin{bmatrix} Y'_{601} \\ U \\ V \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.544639 & 0.838671 \\ 0 & 0.838671 & 0.544639 \end{bmatrix} \bullet \begin{bmatrix} Y'_{601} \\ I \\ Q \end{bmatrix}$$

33 How should I test my encoders and decoders?

To test your encoding and decoding, ensure that colorbars are handled correctly. A colorbar signal comprises a binary RGB sequence ordered for decreasing luma: white, yellow, cyan, green, magenta, red, blue and black.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

To ensure that your scale factors are correct and that clipping is not being invoked, test 75% bars, a colorbar sequence having 75%-amplitude bars instead of 100%.

34 What is perceptual uniformity?

A system is *perceptually uniform* if a small perturbation to a component value is approximately equally perceptible across the range of that value. The volume control on your radio is designed to be perceptually uniform: rotating the knob ten degrees produces approximately the same perceptual increment in volume anywhere across the range of the control. If the control were physically linear, the logarithmic nature of human loudness perception would place all of the perceptual "action" of the control at the bottom of its range.

The XYZ and RGB systems are far from exhibiting perceptual uniformity. Finding a transformation of XYZ into a reasonably perceptually-uniform space consumed a decade or more at the CIE and in the end no single system could be agreed. So the CIE standardized two systems, *L*u*v** and *L*a*b**, sometimes written CIELUV and CIELAB. (The *u* and *v* are unrelated to video *U* and *V*.) Both *L*u*v** and *L*a*b** improve the 80:1 or so perceptual nonuniformity of *XYZ* to about 6:1. Both demand too much computation to accommodate real-time display, although both have been successfully applied to image coding for printing.

Computation of CIE *L*u*v** involves intermediate *u′* and *v′* quantities, where the prime denotes the successor to the obsolete 1960 CIE *u* and *v* system:

$$u' = \frac{4\,X}{X + 15\,Y + 3\,Z}, \qquad v' = \frac{9\,Y}{X + 15\,Y + 3\,Z}$$

First compute u'_n and v'_n for your reference white X_n, Y_n and Z_n. Then compute *u′* and *v′* – and *L** as discussed earlier – for your colors. Finally, compute:

$$u^* = 13\,L^*\left(u' - u'_n\right), \qquad v^* = 13\,L^*\left(v' - v'_n\right)$$

*L*a*b** is computed as follows, for $(X/X_n, Y/Y_n, Z/Z_n) > 0.01$:

$$a^* = 500\left[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right], \qquad b^* = 200\left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}\right]$$

These equations are great for a few spot colors, but no fun for a million pixels. Although it was not specifically optimized for this purpose, the nonlinear *R′G′B′* coding used in video is quite perceptually uniform, and has the advantage of being fast enough for interactive applications.

35 What are *HSB* and *HLS*?

HSB and *HLS* were developed to specify numerical Hue, Saturation and Brightness (or Hue, Lightness and Saturation) in an age when users had to specify colors numerically. The usual formulations of *HSB* and *HLS* are flawed with respect to the properties of color vision. Now that users can choose colors visually, or choose colors related to other media (such as PANTONE), or use perceptually-based systems like *L*u*v** and *L*a*b**, *HSB* and *HLS* should be abandoned.

Here are some of problems of *HSB* and *HLS*. In color selection where "lightness" runs from zero to 100, a lightness of 50 should appear to be half as bright as a lightness of 100. But the usual formulations of *HSB* and *HLS* make no reference to the linearity or nonlinearity of the underlying *RGB*, and make no reference to the lightness perception of human vision.

The usual formulation of *HSB* and *HLS* compute so-called "lightness" or "brightness" as (R+G+B)/3. This computation conflicts badly with the properties of color vision, as it computes yellow to be about six times more luminant than blue with the same "lightness" value (say *L*=50).

HSB and *HSL* are not useful for image computation because of the discontinuity of hue at 360°. You cannot perform arithmetic mixtures of colors expressed in polar coordinates.

Nearly all formulations of *HSB* and *HLS* involve different computations around 60° segments of the hue circle. These calculations introduce visible discontinuities in color space.

Although the claim is made that *HSB* and *HLS* are "device independent", the ubiquitous formulations are based on *RGB* components whose chromaticities and white point are unspecified. Consequently, *HSB* and *HLS* are useless for conveyance of accurate color information.

If you really need to specify hue and saturation by numerical values, rather than *HSB* and *HSL* you should use polar coordinate version of u^* and v^*: h^*_{uv} for hue angle and c^*_{uv} for chroma.

36 What is *truecolor*?

Truecolor is the provision of three separate components for additive red, green and blue reproduction. A high quality true color system provides 8 bits for each of the three components; this is known as *24-bit color*.

A high-quality true color system interposes a lookup table between each component of the framestore and each channel of the display. This makes it possible to use a true color system with either linear or nonlinear coding. In the X Window System, *true color* refers to fixed lookup tables, and *direct color* refers to lookup tables that are under the control of application software.

37 What is *hicolor*?

A *hicolor* system provides 16 bits for each pixel, partitioned into red, green, and blue components. Hicolor is a variant of truecolor, but with an insufficient number of bits to provide photographic quality. The 16 bits may be partitioned as 5 bits for each component (with the extra bit sometimes used to convey transparency), or as 5 bits of red, 6 bits of green, and 5 bits of blue. Hicolor systems usually offer no lookup table at the output of the framebuffer, so the image data is coded like video: The *RGB* components are assumed to have been raised to a power of about 0.5.

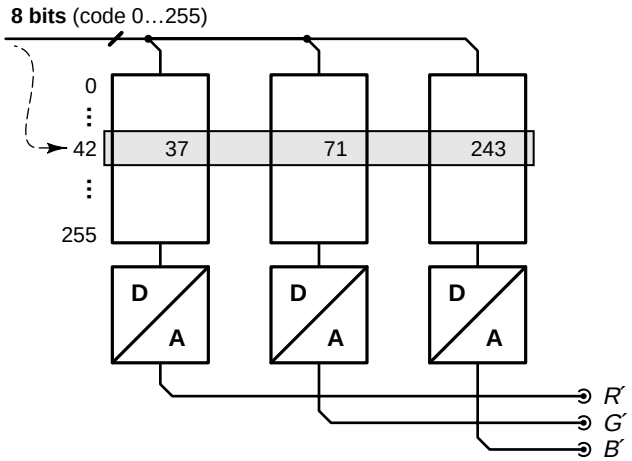


Figure 5 **Pseudocolor (8-bit)** graphics systems are common in low-end personal computers. For each pixel, the framebuffer stores a color index value (typically 8 bits). Each index value is associated with a triplet of *R'G'B'* codes through a mapping loaded into the *color lookup table* (CLUT) of the display hardware. When an index value is accessed from the framebuffer, its triplet is accessed from the CLUT and then applied to the digital-to-analog converter (DAC). *R'G'B'* codes from the CLUT translate linearly into voltage applied to the monitor, so the code values are comparable to video *R'G'B'* values – the *R'G'B'* values are proportional to intensity raised to the 0.45 power.

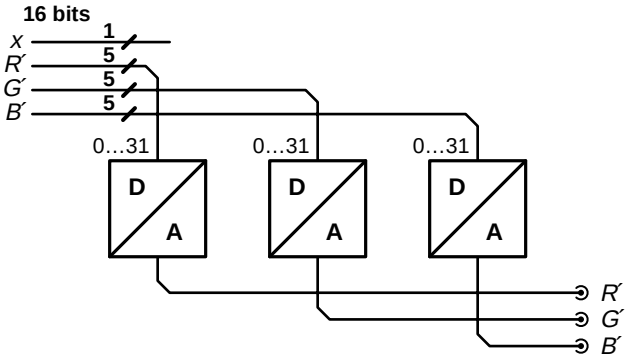


Figure 6 **Hicolor (16-bit)** graphics systems store, in the framebuffer, *R'G'B'* codes partitioned into three components of five bits each (5-5-5), or partitioned five bits for red, six bits for green, and five bits for blue (5-6-5). In low-end systems, these codes are applied directly to the DACs with no intervening colormap. Because the *R'G'B'* codes are translated linearly into monitor voltage, the code values are implicitly proportional to intensity to the 0.45 power, comparable to video *R'G'B'* codes.

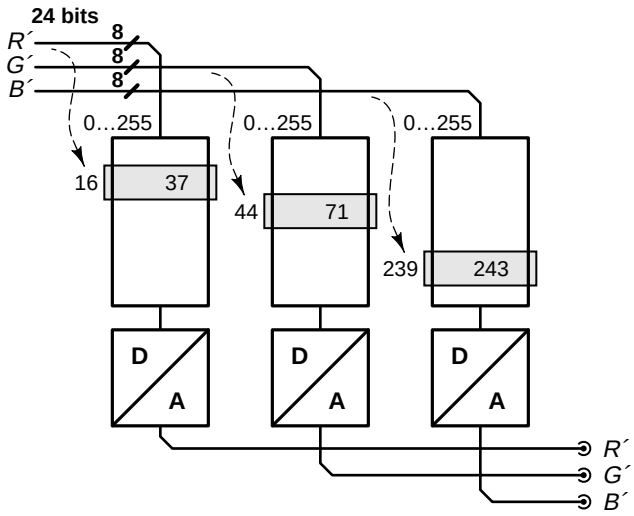


Figure 7 **Truecolor (24-bit)** graphics systems store 8 bits for each of the red, green, and blue components. Truecolor systems usually implement a set of *lookup tables* (LUTs) between the framebuffer memory and the DACs. The LUTs allow a transfer function to be imposed between the *R'G'B'* codes and the DACs: The *R'G'B'* values in the framebuffer need not be related to intensity by the 0.45 power. Application software or system software can impose arbitrary functions. In order for the application software to provide the same default behavior as low-end pseudocolor and hicolor graphics systems, each LUT is set by default to a *ramp* corresponding to the identity (or *unity*) function.

38 What is *indexed color*?

Indexed color (or *pseudocolor)*, is the provision of a relatively small number of discrete colors – often 256 – in a *colormap* or *palette*. The framebuffer stores, at each pixel, the index number of a color. At the output of the framebuffer, a lookup table uses the index to retrieve red, green and blue components that are then sent to the display.

The colors in the map may be fixed systematically at the design of a system. As an example, 216 index entries an eight-bit indexed color system can be partitioned systematically into a 6×6×6 *colorcube* to implement what amounts to a direct color system where each of red, green and blue has a value that is an integer in the range zero to five.

An *RGB* image can be converted to a predetermined colormap by choosing, for each pixel in the image, the colormap index corresponding to the "closest" *RGB* triple. With a systematic colormap such as a 6×6×6 colorcube this is straightforward. For an arbitrary colormap, the colormap has to be searched looking for entries that are "close" to the requested color. "Closeness" should be determined according to the perceptibility of color differences. Using color systems such as CIE *L*u*v** or *L*a*b** is computationally prohibitive, but in practice it is adequate to use a Euclidean distance metric in *R'G'B'* components coded nonlinearly according to video practice.

A direct color image can be converted to indexed color with an image-dependent colormap by a process of color quantization that searches through all of the triples used in the image, and chooses the palette for the image based on the colors that are in some sense most "important". Again, the decisions should be made according to the perceptibility of color differences. Adobe Photoshop™ can perform this conversion. UNIX™ users can employ the *pbm* package.

If your system accommodates arbitrary colormaps, when the map associated with the image in a particular window is loaded into the hardware colormap, the maps associated with other windows may be disturbed. In window system such as the X Window System™ running on a multitasking operating system such as UNIX, even moving the cursor between two windows with different maps can cause annoying *colormap flashing*.

An eight-bit indexed color system requires less data to represent a picture than a twenty-four bit truecolor system. But this data reduction comes at a high price. The truecolor system can represent each of its three components according to the principles of sampled continuous signals. This makes it possible to accomplish, with good quality, operations such as resizing the image. In indexed color these operations introduce severe artifacts because the underlying representation lacks the properties of a continuous representation, even if converted back to *RGB*.

In graphic file formats such as GIF of TIFF, an indexed color image is accompanied by its colormap. Generally such a colormap has *RGB* entries that are gamma corrected: the colormap's *RGB* codes are intended to be presented directly to a CRT, without further gamma correction.

39 I want to visualize a scalar function of two variables. Should I use *RGB* values corresponding to the colors of the rainbow?

When you look at a rainbow you do not see a smooth gradation of colors. Instead, some bands appear quite narrow, and others are quite broad. Perceptibility of hue variation near 540 nm is half that of either 500 nm or 600 nm. If you use the rainbow's colors to represent data, the visibility of differences among your data values will depend on where they lie in the spectrum.

If you are using color to aid in the visual detection of patterns, you should use colors chosen according to the principles of perceptual uniformity. This an open research problem, but basing your system on CIE $L^*a^*b^*$ or $L^*u^*v^*$, or on nonlinear video-like $R'G'B'$, would be a good start.

40 What is dithering?

A display device may have only a small number of choices of grayscale values or color values at each device pixel. However if the viewer is sufficiently distant from the display, the value of neighboring pixels can be set so that the viewer's eye integrates several pixels to achieve an apparent improvement in the number of levels or colors that can be reproduced.

Computer displays are generally viewed from distances where the device pixels subtend a rather large angle at the viewer's eye, relative to his visual acuity. Applying dither to a conventional computer display often introduces objectionable artifacts. However, careful application of dither can be effective. For example, human vision has poor acuity for blue spatial detail but good color discrimination capability in blue. Blue can be dithered across two-by-two pixel arrays to produce four times the number of blue levels, with no perceptible penalty at normal viewing distances.

41 How does halftoning relate to color?

The processes of offset printing and conventional laser printing are intrinsically *bilevel*: a particular location on the page is either covered with ink or not. However, each of these devices can reproduce closely-spaced dots of variable size. An array of small dots produces the perception of light gray, and an array of large dots produces dark gray. This process is called *halftoning* or *screening*. In a sense this is dithering, but with device dots so small that acceptable pictures can be produced at reasonable viewing distances.

Halftone dots are usually placed in a regular grid, although *stochastic screening* has recently been introduced that modulates the spacing of the dots rather than their size.

In color printing it is conventional to use cyan, magenta, yellow and black grids that have exactly the same dot pitch but different carefully-chosen *screen angles*. The technique of stochastic screening uses the same screen angles for all screens, but its registration requirements are more stringent than conventional offset printing.

Agfa's booklet [18] is an excellent introduction to practical concerns of printing. And it's in color! The standard reference to halftoning algorithms is Ulichney [19], but that work does not detail the nonlinearities found in practical printing systems. For details about screening for color reproduction, consult Fink [20]. Consult *Frequently Asked Questions about Gamma* for an introduction to the transfer function of offset printing.

42 What's a color management system?

Software and hardware for scanner, monitor and printer calibration have had limited success in dealing with the inaccuracies of color handling in desktop computing. These solutions deal with specific pairs of devices but cannot address the end-to-end system. Certain application developers have added color transformation capability to their applications, but the majority of application developers have insufficient expertise and insufficient resources to invest in accurate color.

A *color management system* (CMS) is a layer of software resident on a computer that negotiates color reproduction between the application and color devices. It cooperates with the operating system and the graphics library components of the platform software. Color management systems perform the color transformations necessary to exchange accurate color between diverse devices, in various color coding systems including *RGB*, *CMYK* and CIE *L*a*b**.

The CMS makes available to the application a set of facilities whereby the application can determine what color devices and what color spaces are available. When the application wishes to access a particular device, it requests that the color manager perform a mathematical *transform* from one space to another. The color spaces involved can be device-independent *abstract* color spaces such as CIE *XYZ*, CIE *L*a*b** or calibrated *RGB*. Alternatively a color space can be associated with a particular device. In the second case the color manager needs access to characterization data for the device, and perhaps also to *calibration* data that reflects the state of the particular instance of the device.

Apple's ColorSync™ provides an interface between a Mac application program and color management capabilities either built-in to ColorSync or provided by a plug-in. Kodak's CMS is built-into the latest version of Sun's Solaris operating system.

The basic CMS services provided with desktop operating systems are likely to be adequate for office users, but are unlikely to satisfy high-end users such as in prepress. All of the announced systems have provisions for plug-in *color management modules* (CMMs) that can provide sophisticated transform machinery. Advanced color management modules are commercially available from Kodak, Agfa, and others. For an application developer's perspective on color management, see Aldus [21].

43 How does a CMS know about particular devices?

A CMS needs access to information that characterizes the color reproduction capabilities of particular devices. The set of characterization data for a device is called a *device profile*. Industry agreement has been reached on the format of device profiles; information is available from the International Color Consortium (ICC). Vendors of color peripherals will soon provide industry-standard profiles with their devices, and they will have to make, buy or rent characterization services.

If you have a device that has not been characterized by its manufacturer, Agfa's FotoTune™ software – part of Agfa's FotoFlow™ color manager – can create device profiles.

44 Is a color management system useful for color specification?

Not quite yet. But future color management system are likely to include the ability to accommodate commercial proprietary color specification systems such as PANTONE™ and COLORCURVE™. These vendors are likely

to provide their color specification systems in shrink-wrapped form to plug into color managers. In this way, users will have guaranteed color accuracy among applications and peripherals, and application vendors will no longer need to pay to license these systems individually.

45 I'm not a color expert. What parameters should I use to code my images?

Use the CIE D_{65} white point (6504 K) if you can.

Use the Rec. 709 primary chromaticities. Your monitor is probably already quite close to this. Rec. 709 has international agreement, offers excellent performance, and is the basis for HDTV development so it's future-proof.

If you need to operate in linear light, so be it. Otherwise, for best perceptual performance and maximum ease of interchange with digital video, use the Rec. 709 transfer function, with its 0.45-power law. If you need Mac compatibility you will have to suffer a penalty in perceptual performance. Raise tristimulus values to the $1/1.4$-power before presenting them to QuickDraw.

To code luma, use the Rec. 601 luma coefficients 0.299, 0.587 and 0.114. Use Rec. 601 digital video coding with black at 16 and white at 235.

Use prime symbols (′) to denote all of your nonlinear components!

PhotoCD uses all of the preceding measures. PhotoCD codes color differences asymmetrically, according to film gamut. Unless you have a requirement for film gamut, you should code into color differences using $Y'C_BC_R$ coding with Rec. 601 studio video (16..235/128±112) excursion.

Tag your image data with the primary and white chromaticity, transfer function and luma coefficients that you are using. TIFF 6.0 tags have been defined for these parameters. This will enable intelligent readers, today or in the future, to determine the parameters of your coded image and give you the best possible results.

46 References

[1] B. Berlin and P. Kay, *Basic Color Terms* (Berkeley, Calif.: U. of Calif. Press, 1969)

[2] Publication CIE No 17.4, *International Lighting Vocabulary* (Vienna, Austria: Central Bureau of the Commission Internationale de L'Éclairage)

[3] LeRoy E. DeMarsh and Edward J. Giorgianni, "Color Science for Imaging Systems," in *Physics Today*, September 1989, 44-52.

[4] W.F. Schreiber, *Fundamentals of Electronic Imaging Systems, Third Edition* (New York: Springer-Verlag, 1993)

[5] Publication CIE No 15.2, *Colorimetry, Second Edition* (Vienna, Austria: Central Bureau of the Commission Internationale de L'Éclairage, 1986)

[6] Günter Wyszecki and W.S. Styles, *Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition* (New York: Wiley, 1982)

[7] D.B. Judd and Günter Wyszecki, *Color in Business, Science and Industry, Third Edition* (New York: Wiley, 1975)

[8] R.W.G. Hunt, *The Reproduction of Colour, Fifth Edition* (Kingston-upon-Thames, England: Fountain, 1995)

[9] ITU-R Recommendation BT.709, *Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange* (1990), [formerly CCIR Rec. 709] (Geneva: ITU, 1990)

[10] Bruce J. Lindbloom, "Accurate Color Reproduction for Computer Graphics Applications", *Computer Graphics*, Vol. 23, No. 3 (July 1989), 117-126 (proceedings of SIGGRAPH '89)

[11] William B. Cowan, "An Inexpensive Scheme for Calibration of a Colour Monitor in terms of CIE Standard Coordinates", in *Computer Graphics*, Vol. 17, No. 3 (July 1983), 315-321.

[12] SMPTE RP 177-1993, *Derivation of Basic Television Color Equations*.

[13] *Television Engineering Handbook, Featuring HDTV Systems, Revised Edition* by K. Blair Benson, revised by Jerry C. Whitaker (New York: McGraw-Hill, 1992). This supersedes the Second Edition.

[14] Roy Hall, *Illumination and Color in Computer Generated Imagery* (Springer-Verlag, 1989)

[15] Chet S. Haase and Gary W. Meyer, "Modelling Pigmented Materials for Realistic Image Synthesis", in *ACM Transactions on Graphics*, Vol. 11, No. 4, 1992, p. 305.

[16] Maureen C. Stone, William B. Cowan, and John C. Beatty, "Color Gamut Mapping and the Printing of Digital Color Images", in *ACM Transactions on Graphics*, Vol. **7**, No. 3, October 1988.

[17] Charles Poynton, *A Technical Introduction to Digital Video* (New York: Wiley, 1996)

[18] Agfa Corporation, *An introduction to Digital Color Prepress, Volumes 1 and 2* (Mt.Prospect, Ill.: Prepress Education Resources, 800 395 7007, 1990)

[19] Robert Ulichney, *Digital Halftoning* (Cambridge, Mass.: MIT Press, 1988)

[20] Peter Fink, *PostScript Screening: Adobe Accurate Screens* (Mountain View, Calif.: Adobe Press, 1992)

[21] *Color management systems: Getting reliable color from start to finish*, Adobe Systems, <http://www.adobe.com/PDFs/FaxYI/500301.pdf>.

[22] *Overview of color publishing*, Adobe Systems, <http://www.adobe.com/PDFs/FaxYI/500302.pdf>. Despite appearances and title, this document is in grayscale, not color.

47 Contributors

Thanks to Norbert Gerfelder, Alan Roberts and Fred Remley for their proofreading and editing. I learned about color from LeRoy DeMarsh, Ed Giorgianni, Junji Kumada and Bill Cowan. Thanks!

This note contains some errors: if you find any, please let me know.

I welcome suggestions for additions and improvements.