How to Derive a Spectrum From an RGB Triple

Andrew Glassner

Faculty of Mathematics and Informatics
Delft University of Technology
Julianalaan 132
2628 BL Delft
The Netherlands

Technical Memo #87-2
27 November 1987

Abstract

Accurate display of synthetic images requires careful attention to color. Physical phenomena such as reflection and transmission are wavelength-dependent, so these effects should be modelled for the entire visible light spectrum. It can be shown that simply modelling this spectrum with an RGB triplet leads to color aliasing [HALL84]. One apparent hurdle in supporting the spectral model in rendering systems is that many users have built up libraries of colors based on RGB triplets which they are loath to abandon. In this note we present a simple method that takes as input an RGB triplet viewed on a given monitor, and computes a corresponding spectrum for use in the renderer. The method can also find a spectrum for colors described as HSV triplets, and xy pairs or xyz triplets in CIE XYZ color space.

1.0 Problem Context and Statement

The problem addressed in this note involves support for a rendering system that does color computation on a finely (and perhaps stochastically) sampled color spectrum. Pictures produced by such a system consist of a color spectrum for each pixel in the screen viewport. For display on a particular color monitor each pixel's spectrum must be converted into an RGB triple. To find the RGB triple for a given spectrum, the system first calculates which point in XYZ color space is associated with that spectrum. That XYZ point is then processed by the XYZ-to-RGB transformation for the appropriate monitor, which accounts for the particular phosphors in that tube.

We wish to allow users of such a spectrum-based system to specify colors as RGB triples seen on a particular monitor. More precisely, we wish to accept from the user an RGB triple viewed on a particular monitor, and from the RGB values and the monitor description find a spectrum that will generate, after the above conversion process from spectrum to XYZ color space to the monitor gamut, the same RGB triplet.

Such a capability allows a designer to choose RGB colors interactively on a certain monitor, and then assign those colors to objects in a synthetic scene. Such colors might describe reflected or emitted spectra. If a color is used as a diffuse reflection spectrum, then a (purely diffuse) object illuminated by unit white light in a rendered scene will be perceived as having the same color as the original RGB triplet, no matter which display device is used.

Our technique assumes that all involved devices are gamma-corrected, so that the relationship between RGB signal and perceived brightness can be considered linear [CATMULL79].

2.0 Notation for XYZ $\leftrightarrow$ RGB Transformations

Consider the standard conversion of a color in CIE XYZ space to a particular monitor's RGB space. This is accomplished via a linear transformation $\mathbf{M}_i$, built for monitor i. It is useful to write the XYZ color as a position vector $\mathbf{X} = [x\ y\ z]$, and the RGB color as a position vector $\mathbf{R} = [r\ g\ b]$ (where $0 \le x, y, z, r, g, b \le 1$). Then $\mathbf{M}_i$ is a 3-by-3 matrix built from the chromaticities of the phosphors in monitor i [HALL86], which converts a color in XYZ space to the corresponding RGB of that (gamma-corrected) monitor. We thus have the relation

(1) $\mathbf{R} = \mathbf{X}\mathbf{M}_i$

2.1 Notation for XYZ ↔ Spectrum Transformations

To convert a spectrum A into XYZ space we integrate with the three CIE standard matching functions $\bar{x}$, $\bar{y}$, and $\bar{z}$ to produce the values x, y, and z:

$$x = \int_{380}^{780} A(\lambda)\ \bar{x}(\lambda)\ d\lambda \qquad y = \int_{380}^{780} A(\lambda)\ \bar{y}(\lambda)\ d\lambda \qquad z = \int_{380}^{780} A(\lambda)\ \bar{x}(\lambda)\ d\lambda$$

Consider the infinite number of spectra A that give the same XYZ co-ordinates. We're free to choose any such spectrum, since one of Grassman's laws [GRASSMAN1854] is that "stimuli of the same color (that is, matched by the same amounts of the same three light sources) produce identical effects in mixtures regardless of their spectral composition" (quoted from [MEYER]). Since the transformation from XYZ to RGB is linear, all spectra which give us the desired RGB also give the same XYZ. So we may pick any spectrum we like, as long as it gives the proper XYZ to get the desired RGB.

3.0 Solution

We define **T** to be a 3-dimensional space of line spectra (a line spectrum is 0 everywhere but at some finite number of wavelengths), as illustrated in Figure 1. Each axis of **T** corresponds to a line spectrum that is 0 everywhere except for unit amplitude at one fixed wavelength; i.e. the axes u, v, and w of T correspond to spectra T_u, T_v, T_w that are 0 everywhere except $T_u(\lambda_u)=1$, $T_v(\lambda_v)=1$, and $T_w(\lambda_w)=1$. Each point S in T corresponds to a spectrum S that is a linear combination of these spectra, so it is 0 everywhere except at these three wavelengths, where the amplitude of the spectrum is equal to the associated co-ordinate of S. Thus a point S = [a b c] corresponds to a spectrum S that is zero everywhere, except $S(\lambda_u)=a$, $S(\lambda_v)=b$, and $S(\lambda_w)=c$.

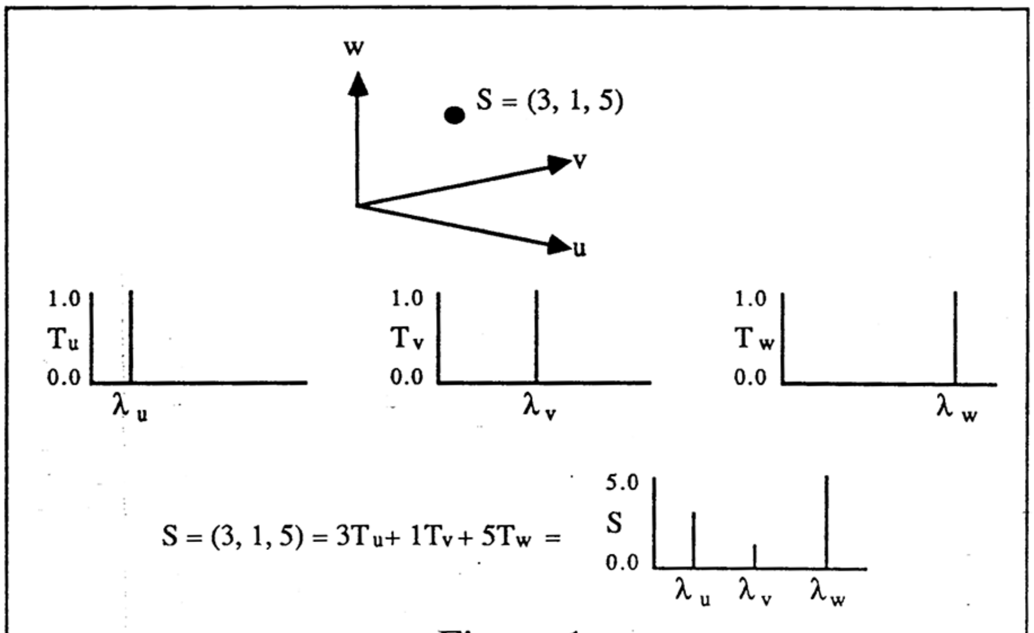

Figure 1

For such a spectrum S we can directly write the integrals in Section 2.1:

$$x = S(\lambda_u)\,\bar{x}(\lambda_u) + S(\lambda_v)\,\bar{x}(\lambda_v) + S(\lambda_w)\,\bar{x}(\lambda_w)$$
$$y = S(\lambda_u)\,\bar{y}(\lambda_u) + S(\lambda_v)\,\bar{y}(\lambda_v) + S(\lambda_w)\,\bar{y}(\lambda_w)$$
$$z = S(\lambda_u)\,\bar{z}(\lambda_u) + S(\lambda_v)\,\bar{z}(\lambda_v) + S(\lambda_w)\,\bar{z}(\lambda_w)$$

We may write the above equations as a product of the vector S and a matrix C, where C holds the nine entries corresponding to the amplitudes of $\bar{x}$, $\bar{y}$, and $\bar{z}$ at the three wavelengths λ_u, λ_v, and λ_w. We thus have our second relation:

$$(2) \qquad \mathbf{X} = \mathbf{SC}$$

$$\text{where } \mathbf{S} = [S(\lambda_u)\ S(\lambda_v)\ \ S(\lambda_w)]\,,\ \ \mathbf{C} = \begin{bmatrix} \bar{x}(\lambda_u) & \bar{y}(\lambda_u) & \bar{z}(\lambda_u) \\ \bar{x}(\lambda_v) & \bar{y}(\lambda_v) & \bar{z}(\lambda_v) \\ \bar{x}(\lambda_w) & \bar{y}(\lambda_w) & \bar{z}(\lambda_w) \end{bmatrix}$$

We now have everything we need to find the spectrum S corresponding to to an RGB vector **R**. Equation 1 relates points **R** in RGB color space to points **X** in XYZ color space, and Equation 2 relates those points **X** to points S in the spectrum space. The relation between **R** and S is just the composite transformation $\mathbf{R} = \mathbf{XM}_i = (\mathbf{SC})\mathbf{M}_i = \mathbf{S}(\mathbf{CM}_i)$. Solving for S gives our solution:

$$\mathbf{S} = \mathbf{R}(\mathbf{CM}_i)^{-1}$$

To get from a given XYZ triple to a spectrum is even easier; just solve (2) for $\mathbf{S} = \mathbf{XC}^{-1}$.

4.0 Discussion

Matrix inversion is a costly process. If we compute $\mathbf{J} = (\mathbf{CM}_i)^{-1}$ the first time we convert a color to a spectrum and save **J**, then successive color transformations will only require a matrix multiply. We can eliminate even this one run-time inversion under some circumstances.

Note that matrix **C** is a function of the CIE standard observer curves, which don't change. Below we discuss how to build a good **C** matrix, but once built it will remain fixed. On the other hand, $\mathbf{M}_i$ is dependent on display device i. If we allow the user to specify the device characteristics at run time, then we'll have to build the appropriate $\mathbf{M}_i$ on the fly, compose it with C, and compute **J**. Alternatively, we can keep a file of all the different displays in a given lab, from which the user selects one display device. Associated with each display in the file is the transformation matrix **J**, which we read in directly.

If we have 4-by-4 matrix routines available (e.g. if another part of the program does spatial transformations), then we can use those routines without modification if we must build **J** at run time. Just fill out each matrix and vector with identity homogeneous co-ordinates and implement the equations from section 3.

A useful feature is to provide spectrum generation for HSV color descriptions [SMITH78]; simply read in an HSV color, convert it to RGB, and run the RGB triplet through **J** to get the corresponding spectrum.

It may be interesting to consider the physical significance of **J**. A point in RGB space represents a linear sum of the three RGB basis vectors Red, Blue, and Green. Similarly, points in the spectrum space (which we associate with line spectra above) are linear sums of the spectrum space bases, each of which is a one-component line spectrum. So the weights on the Red, Green, and Blue axes, which together make an RGB color, are transformed into weights for each of the three line spectra, which together make the associated 3-component line spectrum.

The spectra of the axes of the spectrum space **T** are non-zero only at three particular wavelengths: λ_u, λ_v, and λ_w. We must exercise a bit of caution when picking these wavelengths, since they give rise to our C matrix, which must be invertible. Thus, the three wavelengths must be different (to avoid linear dependency), and all three matching functions must be non-zero at all three wavelengths (to avoid a zero determinant). To get the best numerical stability when inverting the matrix choose the wavelengths that give the largest (but not too similar) values for the matrix. I associate λ_u, λ_v, and λ_w with the wavelength of the largest entry in my sampled versions of the $\bar{x}$, $\bar{y}$, and $\bar{z}$ matching functions respectively; it fulfills the matrix criteria and seems to be numerically stable. I present my matrix **C** and the wavelengths from which it was computed in the Appendix.

Observe that the technique in this note does not select **the** spectrum corresponding to a given RGB triple; an infinite number of such spectra exist. In fact, the spectrum built from just a single application of this method is probably one of the worst for the job, since it is zero everywhere but at three specific wavelengths. In a situation where this spectrum describes a transparent filter, most of the light energy will be blocked; a more spread-out spectrum would probably pass more energy.

One way to get a fuller spectrum is to build several spectra by the above process. Each spectrum is built using a different set of wavelengths as the basis for the spectrum space **T** (as defined in Section 3.0). For each set n, $(\lambda_u, \lambda_v, \lambda_w)_n$ defines a different space $\mathbf{T}_n$, and thus requires its own transformation matrix $\mathbf{J}_n$. Once we have computed each of these spectra, we average them together (we can use any weights on the spectra when averaging; equal weighting works fine). Again using Grassman's law we know that the composite, averaged spectrum will produce the same effect as any of its components.

5.0 References

[CATMULL79] Catmull, E., "A Tutorial on Compensation Tables", *Computer Graphics* **13** (3), Proceedings of Siggraph '79

[GRASSMAN1854] Grassman, H., "On the Theory of Compound Colors," *Phil. Mag*, April 1854

[HALL84] Hall, R., and D. Greenberg, "A Testbed for Realistic Image Synthesis", *IEEE Computer Graphics & Applications*, vol. 3, no. 8, November 1983

[HALL86] Hall, R., "Color Reproduction and Illumination Models," *Techniques for Computer Graphics*, Editors D. Rogers and R. Earnshaw, Springer-Verlag 1987

[MEYER] Meyer, G., and D. Greenberg, "Colorimetry and Computer Graphics," Program of Computer Graphics, Cornell University, Ithaca, NY, unpublished.

[SMITH78] Smith, A.R., "Color Gamut Transform Pairs," *Computer Graphics* **12** (3), Proceedings of Siggraph '78

APPENDIX

My matrix C encoding the spectrum-to-XYZ transformation is built from samples of the matching functions at $\lambda_u = 590$nm, $\lambda_v = 560$nm, and $\lambda_w = 440$nm. The corresponding entries of the $\bar{x}$, $\bar{y}$, and $\bar{z}$ matching functions then give the C below. Here are also my default $\mathbf{M_i}$ and the resulting $\mathbf{J} = (\mathbf{CM_i})^{-1}$:

$$\mathbf{C} = \begin{bmatrix} 1.0263 & 0.7570 & 0.0011 \\ 0.5945 & 0.9950 & 0.0039 \\ 0.3483 & 0.0230 & 1.7471 \end{bmatrix}, \qquad \mathbf{C}^{-1} = \begin{bmatrix} 1.7411 & -1.3247 & 0.0019 \\ -1.0389 & 1.7955 & -0.0034 \\ -0.3334 & 0.2404 & 0.5721 \end{bmatrix}$$

$$\mathbf{M_i} = \begin{bmatrix} 2.1336 & -1.1279 & 0.0103 \\ -0.6882 & 2.0517 & -0.1568 \\ -0.3421 & 0.0463 & 0.9689 \end{bmatrix}, \quad \mathbf{J} = (\mathbf{CM_i})^{-1} = \begin{bmatrix} 0.6623 & -0.1861 & 0.0258 \\ -0.2916 & 0.8236 & 0.0565 \\ -0.0963 & 0.1431 & 0.5970 \end{bmatrix}$$

Examples: xy→S and RGB→S

Let us take as an example an input in xy chromaticity space (.1578, .2528). We compute the z value as 1-x-y=.5894 so $\mathbf{X} = [0.1578\ 0.2528\ 0.5894]$. Then $\mathbf{S} = \mathbf{XC}^{-1} = [-0.1844\ \ 0.3866\ \ 0.3366]$. So the spectrum S giving this **X** is zero everywhere except $S_{\lambda=590} = -0.1844$, $S_{\lambda=560} = 0.3866$, $S_{\lambda=440} = 0.3366$ (note the negative intensity!).

As a second example let us take the RGB input vector $\mathbf{R} = [.0191, .9967, .7749]$. Then the composite transformation gives us $\mathbf{S} = \mathbf{R}(\mathbf{CM_i})^{-1} = \mathbf{RJ} = [-0.3526\ 0.9283\ 0.5156]$.

Computing the XYZ-to-RGB Transformation Matrix

For completeness, we give a summary of how to derive the XYZ-to-RGB transformation matrix $\mathbf{M_i}$ for a given monitor. A more complete discussion may be found in [HALL86].

From the monitor specifications we get the xy chromaticities for the CRT white spot (wx, wy) and red, green, and blue phosphors (rx, ry), (gx, gy), and (bx, by). We augment each xy pair with the corresponding z = 1-x-y. From the phosphor triplets we build the matrix **K** below. From the white spot triplet we build the XYZ color vector **W**, which is the color corresponding to the chromaticities (wx, wy) with the luminance Y set to 1.0. We will find it useful in the derivation to define the RGB white vector $\mathbf{F} = [1\ 1\ 1]$, and the matrix $\mathbf{N_i} = (\mathbf{M_i})^{-1}$. We also define the vector **V** and matrix **G**, which are related by $\mathbf{V} = \mathbf{FG}$.

$$\mathbf{W} = \begin{bmatrix} \frac{wx}{wy} & 1 & \frac{w\,z}{wy} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} r_x & r_y & r_z \\ g_x & g_y & g_z \\ b_x & b_y & b_z \end{bmatrix}, \quad \mathbf{V} = [G_x\ G_y\ G_z], \quad \mathbf{G} = \begin{bmatrix} G_x & 0 & 0 \\ 0 & G_y & 0 \\ 0 & 0 & G_z \end{bmatrix}$$

First, we observe $\mathbf{N_i} = \mathbf{GK}$ (the RGB-to-XYZ matrix is a differentially scaled version of the phosphor matrix). Since $\mathbf{N_i}$ takes the XYZ white **W** to the RGB white **F**, we write $\mathbf{W} = \mathbf{FN_i}$. Substituting for $\mathbf{N_i}$, $\mathbf{W} = \mathbf{F(GK)}$. Multiplying by $\mathbf{K}^{-1}$ on both sides gives $\mathbf{WK}^{-1} = \mathbf{FG}$. Recalling $\mathbf{V} = \mathbf{FG}$, we can rewrite this last equation as $\mathbf{V} = \mathbf{WK}^{-1}$ and thereby compute **V**. From **V** build the corresponding matrix **G**. Then compute $\mathbf{N_i} = \mathbf{GK}$, and from that compute $\mathbf{M_i} = (\mathbf{N_i})^{-1}$. In summary, the steps are:

1. Build **W** and **K** from the monitor white spot and color phosphors.
2. Compute $\mathbf{V} = \mathbf{WK}^{-1}$
3. From the vector **V** build the matrix **G**
4. Compute $\mathbf{N_i} = \mathbf{GK}$
5. Compute $\mathbf{M_i} = (\mathbf{N_i})^{-1}$

The $\mathbf{M_i}$ given at the start of this Appendix is the result of the above procedure for a monitor with phosphors (rx, ry) = (.615, .337), (gx, gy) = (.231, .664), (bx, by) = (.147, .063), and white spot (wx, wy) = (.310, .316).