High Dynamic Range Imaging

Greg Ward
Exponent – Failure Analysis Assoc.
Menlo Park, California

Abstract

The ultimate in color reproduction is a display that can produce arbitrary spectral content over a 300-800 nm range with 1 arc-minute resolution in a full spherical hologram. Although such displays will not be available until next year, we already have the means to calculate this information using physically-based rendering. We would therefore like to know: how may we represent the results of our calculation in a device-independent way, and how do we map this information onto the displays we currently own? In this paper, we give an example of how to calculate full spectral radiance at a point and convert it to a reasonably correct display color. We contrast this with the way computer graphics is usually done, and show where reproduction errors creep in. We then go on to explain reasonable short-cuts that save time and storage space without sacrificing accuracy, such as illuminant discounting and human gamut color encodings. Finally, we demonstrate a simple and efficient tone-mapping technique that matches display visibility to the original scene.

Introduction

Most computer graphics software works in a 24-bit RGB space, with 8-bits allotted to each of the three primaries in a power-law encoding. The advantage of this representation is that no tone-mapping is required to obtain a reasonable reproduction on most commercial CRT display monitors, especially if both the monitor and the software adhere to the sRGB standard, i.e., CCIR-709 primaries and a 2.2 gamma [1]. The disadvantage of this practice is that colors outside the sRGB gamut cannot be represented, particularly values that are either too dark or too bright, since the useful dynamic range is only about 90:1, less than 2 orders of magnitude. By contrast, human observers can readily perceive detail in scenes that span 4-5 orders of magnitude in luminance through local adaptation, and can adapt in minutes to over 9 orders of magnitude. Furthermore, the sRGB gamut only covers about half the perceivable colors, missing large regions of blue-greens and violets, among others. Therefore, although 24-bit RGB does a reasonable job of representing what a CRT monitor can display, it does a poor job representing what a human observer can see.

Display technology is evolving rapidly. Flat-screen LCD displays are starting to replace CRT monitors in many offices, and LED displays are just a few years off. Micromirror projection systems with their superior dynamic range and color gamut are already widespread, and laser raster projectors are on the horizon. It is an important question whether we will be able to take full advantage and adapt our color models to these new devices, or will we be limited as we are now to remapping sRGB to the new gamuts we have available -- or worse, getting the colors wrong? Unless we introduce new color models to our image *sources* and do it soon, we will never get out of the CRT color cube.

The simplest solution to the gamut problem is to adhere to a floating-point color space. As long as we permit values greater than one and less than zero, any set of color primaries may be linearly transformed into any other set of color primaries without loss. The principal disadvantage of most floating-point representations is that they take up too much space (96-bits/pixel as opposed to 24). Although this may be the best representation for color computations, storing this information to disk or transferring it over the internet is a problem. Fortunately, there are representations based on human perception that are compact and sufficiently accurate to reproduce any visible color in 32-bits/pixel or less, and we will discuss some of these in this paper.

There are two principal methods for generating high dynamic-range source imagery: physically-based rendering (e.g., [2]), and multiple-exposure image capture (e.g., [3]). In this paper, we will focus on the first method, since it is most familiar to the author. It is our hope that in the future, camera manufacturers will build HDR imaging principles and techniques into their cameras, but for now, the easiest path to full gamut imagery seems to be computer graphics rendering.

Computer graphics lifts the usual constraints associated with physical measurements, making floating-point color the most natural medium in which to work. If a renderer is physically-based, it will compute color values that correspond to spectral radiance at each point in the rendered image. These values may later be converted to displayable colors, and the how and wherefore of this tone-mapping operation is the main topic of this paper. Before we get to tone-mapping, however, we must go over some of the details of physically-based rendering, and what qualifies a renderer in this category. Specifically, we will detail the basic lighting calculation, and compare this to common practice in computer graphics rendering. We highlight some common assumptions and approximations, and describe alternatives when these assumptions fail. Finally, we demonstrate color and tone mapping methods for converting the computed spectral radiance value to a displayable color at each pixel.

The Spectral Rendering Equation

$$R_o(\omega_o,\lambda) = \iint f_r(\omega_o;\omega_i,\lambda)\, R_i(\omega_i,\lambda)\cos\theta_i \, d\omega_i \quad (1)$$

The spectral rendering Eq. (1) expresses outgoing spectral radiance R_o at a point on a surface in the direction ω_o (θ_o,ϕ_o) as a convolution of the bidirectional reflectance distribution function (BRDF) with the incoming spectral radiance over the projected hemisphere. This equation is the basis of many physically-based rendering programs, and it already contains a number of assumptions:

1. Light is reflected at the same wavelength at which it is received; i.e., the surface is not fluorescent.
2. Light is reflected at the same position at which it is received; i.e., there is no subsurface scattering.
3. Surface transmission is zero.
4. There are no polarization effects.
5. There is no diffraction.
6. The surface does not spontaneously emit light.

In general, these assumptions are often wrong. Starting with the first assumption, many modern materials such as fabrics, paints, and even detergents, contain "whitening agents" which are essentially phosphors added to absorb ultraviolet rays and re-emit them at visible wavelengths. The second assumption is violated by many natural and man-made surfaces, such as marble, skin, and vinyl. The third assumption works for opaque surfaces, but fails for transparent and thin, translucent objects. The fourth assumption fails for any surface with a specular (shiny) component, and becomes particularly troublesome when skylight (which is strongly polarized) or multiple reflections are involved. The fifth assumption fails when surface features are on the order of the wavelength of visible light, and the sixth assumption is violated for light sources.

Each of these assumptions may be addressed and remedied as necessary. Since a more general rendering equation would require a long and tedious explanation, we merely describe what to add to account for the effects listed. To handle fluorescence, the outgoing radiance at wavelength λ_o may be computed from an integral of incoming radiance over all wavelengths λ_I, which may be discretized in a matrix form [4]. To handle subsurface scattering, we can integrate over the surface as well as incoming directions, or use an approximation [5]. To handle transmission, we simply integrate over the sphere instead of the hemisphere, and take the absolute value of the cosine for the projected area [2]. To account for polarization, we add two terms for the transverse and parallel polarizations in each specular direction [4] [6]. To handle diffraction, we fold interactions between wavelength, polarization, amplitude and direction into the BRDF and the aforementioned extensions [7]. Light sources are the simplest exception to handle – we simply add in the appropriate amount of spontaneous radiance output as a function of direction and wavelength.

Participating Media

Implicitly missing from Eq. (1) is the interaction of light with the atmosphere, or participating media. If the space between surfaces contains significant amounts of dust, smoke, or condensation, a photon leaving one surface may be scattered or absorbed along the way. An additional equation is therefore needed to describe this volumetric effect, since the rendering equation only addresses interactions at surfaces.

$$\frac{dR(s)}{ds} = -\sigma_a R(s) - \sigma_s R(s) + \frac{\sigma_s}{4\pi}\int R_i(\theta_i) P(\theta_i)\, d\omega \quad (2)$$

Eq. (2) gives the differential change in radiance as a function of distance along a path. The coefficients σ_a and σ_s give the absorption and scattering densities respectively at position s, which correspond to the probabilities that light will be absorbed or scattered per unit of distance traveled. The scattering phase function, $P(\theta_i)$, gives the relative probability that a ray will be scattered in from direction θ_i at this position. All of these functions and coefficients are also a function of wavelength.

The above differential-integral equation is usually solved numerically by stepping through each position along the path, starting with the radiance leaving a surface given by Eq. (1). Recursive iteration from a sphere of scattered directions can quickly overwhelm such a calculation, especially if it is extended to multiple scattering events. Without going into details, Rushmeier et al. approached the problem of globally participating media using a zonal approach akin to radiosity that divides the scene into a finite set of voxels whose interactions are characterized in a form-factor matrix [8]. More recently, a modified ray-tracing method called the photon map has been applied successfully to this problem by Wann Jensen et al. [9]. In this method, photons are tracked as they scatter and are stored in the environment for later resampling during rendering .

Solving the Rendering Equation

Eq. (1) is a Fredholm integral equation of the second kind, which comes close to the appropriate level of intimidation but fails to explain why it is so difficult to solve in general [10]. Essentially, the equation defines outgoing radiance as an integral of incoming radiance at a surface point, and that incoming radiance is in turn defined by the same integral with different parameters evaluated at another surface point. Thus, the surface geometry and material functions comprise the boundary conditions of an infinitely recursive system of integral equations. In some sense, it is remarkable that researchers have made any progress in this area at all, but in fact, there are many people in computer graphics who believe that rendering is a solved problem.

For over fifteen years, three approaches have dominated research and practice in rendering. The first approach is usually referred to as the *local illumination* approximation, and is the basis for most graphics rendering hardware, and

much of what you see in movies and games. In this approximation, the integral equation is converted into a simple sum over light sources (i.e., concentrated emitters) and a general ambient term. The second approach is called *ray tracing*, and as its name implies, this method traces additional rays to determine specular reflection and transmission, and may be used to account for more general interreflections as well [11] [12]. The third approach is called *radiosity* after the identical method used in radiative transfer, where reflectances are approximated as Lambertian and the surfaces are divided into patches to convert the integral equation into a large linear system that may be solved iteratively [13]. Comparing these three approaches, local illumination is the cheapest and least accurate. Ray tracing has the advantage of coping well with complex geometry and materials, and radiosity does the best job of computing global interactions in simpler, diffuse environments.

In truth, none of the methods currently in use provides a complete and accurate solution to the rendering equation for general environments, though some come closer than others. The first thing to recognize in computer graphics, and computer simulation in general, is that the key to getting a reasonable answer is finding the right approximation. The reason that local illumination is so widely employed when there are better techniques available is not simply that it's cheaper; it provides a reasonable approximation to much of what we see. With a few added tricks, such as shadow maps, reflection maps and ambient lights, local illumination in the hands of an expert does a very credible job. However, this is not to say that the results are correct or accurate. Even in perceptual terms, the colors produced at each pixel are usually quite different from those one would observe in a real environment. In the entertainment industry, this may not be a concern, but if the application is prediction or virtual reenactment, better accuracy is necessary.

For the remainder of this paper, we assume that accuracy is an important goal, particularly color accuracy. We therefore restrict our discussion of rendering and display to physically-based global illumination methods, such as ray-tracing and radiosity.

Tone Mapping

By computing an approximate solution to Eq. (1) for a given planar projection, we obtain a spectral rendering that represents each image point in physical units of radiance per wavelength (e.g., SI units of watts/steradian/meter^2/nm). Whether we arrive at this result by ray-tracing, radiosity, or some combination, the next important task is to convert the spectral radiances to pixel color values for display. If we fail to take this step seriously, it almost doesn't matter how much effort we put into the rendering calculation – the displayed image will look wrong.

Converting a spectral image to a display image is usually accomplished in two stages. The first stage is to convert the spectral radiances to a tristimulus space, such as CIE XYZ. This is done by convolving each radiance spectrum with the three standard CIE observer functions. The second stage is to map each tristimulus value into our target display's color space. This process is called *tone-mapping*, and depending on our goals and requirements, we may take different approaches to arrive at different results. Here are a few possible *rendering intents*:

1. Colorimetric intent: Attempt to reproduce the exact color on the display, ignoring viewer adaptation.[1]
2. Saturation intent: Maintain color saturation as far as possible, allowing hue to drift.
3. Perceptual intent: Attempt to match perception of color by remapping to display gamut and viewer adaptation.

The rendering intents listed above have been put forth by the ICC profile committee, and their exact meaning is somewhat open to interpretation, especially for out-of-gamut colors. Even for in-gamut colors, the perceptual intent, which interests us most, may be approached in several different ways. Here are a few possible techniques:

A. Shrink the source (visible) gamut to fit within the display gamut, scaling uniformly about the neutral line.
B. Same as A, except apply relative scaling so less saturated colors are affected less than more saturated ones. The extreme form of this is gamut-clipping.
C. Scale colors on a curve determined by image content, as in a global histogram adjustment.
D. Scale colors locally based on image spatial content, as in Land's retinex theory.

To any of the above, we may also add a white point transformation and/or contrast adjustment to compensate for a darker or brighter surround. In general, it is impossible to reproduce exactly the desired observer stimulus unless the source image contains no bright or saturated colors or the display has an unusually wide gamut and dynamic range.[2]

Before we can explore any gamut-mapping techniques, we need to know how to get from a spectral radiance value to a tristimulus color such as XYZ or RGB. The calculation is actually straightforward, but the literature on this topic is vast and confusing, so we give an explicit example to make sure we get it right.

Correct Color Rendering

Looking at the simplest case, spectral reflection of a small light source from a diffuse surface in Eq. (1) reduces to the following formula for outgoing radiance:

$$R_o(\lambda) = \frac{\rho_d(\lambda)}{\pi} E_i(\lambda) \qquad (3)$$

[1] The ICC Colorimetric intent is actually divided into *relative* and *absolute* intents, but this distinction is irrelevant to our discussion.

[2] See www.hitl.washington.edu/research/vrd/ for information on Virtual Retinal Display technology.

where $\rho_d(\lambda)$ is the diffuse reflectance as a function of wavelength, and $E_i(\lambda)$ is the spectral irradiance computed by integrating radiance over the projected source. To convert this to an absolute XYZ color, we apply the standard CIE conversion, given below for SI units [16]:

$$X = 683 \int \bar{x}(\lambda) R(\lambda) d\lambda$$
$$Y = 683 \int \bar{y}(\lambda) R(\lambda) d\lambda$$
$$Z = 683 \int \bar{z}(\lambda) R(\lambda) d\lambda \qquad (4)$$

At this point, we may wish to convert to an opponent color space for gamut-mapping, or we may wait until we are in the device color space. If our tone-mapping is a simple scale factor as described in technique A above, we may apply it in any linear color space and the results will be the same. If we convert first to a nonlinear device color space, we need to be aware of the meaning of out-of-gamut colors in that space before we map them back into the legal range of display values. We demonstrate a consistent and reasonable method, then compare to what is usually done in computer graphics.

BlueFlower Example

To compute the absolute CIE color for a surface point, we need to know the spectra of the source and the material. Fig. 1 shows the source spectra for standard illuminant A (2856K tungsten), illuminant B (simulated sunlight), and illuminant D65 (6500K daylight). Fig. 2 shows the reflected spectral radiance of the BlueFlower patch from the MacBeth chart under each of these illuminants. To these curves, we apply the CIE standard observer functions using Eq. (4).

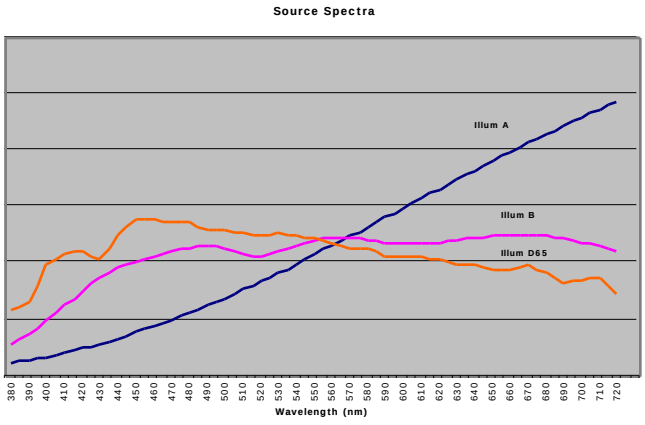

Figure 1. Spectral power of three standard illuminants.

The resulting XYZ values for the three source conditions is given in the first row of Table 1. Not surprisingly, there is a large deviation in color under different illuminants, especially tungsten. We can convert these colors to their RGB equivalents using Eq. (5), as given in the second row of Table 1. If we were to directly display the colors from the illuminant A and B conditions on the screen, they would likely appear incorrect because the viewer would be adapted to the white point of the monitor rather than the white point of the original scenes being rendered. If we assume the scene white point is the same color as the illuminant and the display white point is D65, then a white point adjustment is necessary for the other illuminants (A and B), as given in the third row of Table 1.

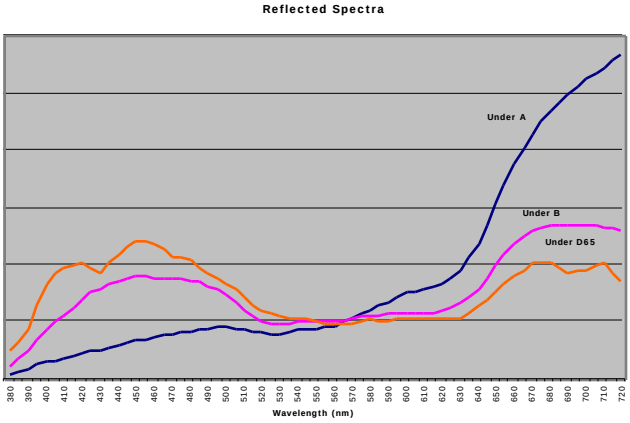

Figure 2. Spectral radiance of MacBeth BlueFlower patch under three standard illuminants.

Source CIE (x,y)	Illum D65 (.3127,.3290)	Illum B (.3484,.3516)	Illum A (.4475,.4075)
BlueFlower CIE XYZ	0.274 0.248 0.456	0.280 0.248 0.356	0.302 0.248 0.145
709 RGB (absolute)	0.279 0.219 0.447	0.349 0.209 0.341	0.525 0.179 0.119
709 RGB (adjusted)	0.279 0.219 0.447	0.285 0.218 0.444	0.306 0.215 0.426

Table 1. Computed color values for BlueFlower under three standard illuminants.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \mathbf{C} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$
$$\mathbf{C}_{709} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix} \qquad (5)$$

We use a linear transform to adjust the white point from that of the illuminant to that of the display, which we assume to be D65 in this example. Eq. (5) gives the absolute transformation from XYZ to CCIR-709 linear RGB, and

this is all we need for the D65 illuminant condition. For the others, we apply the transformation shown in Eq. (6).

Eq. (6) is the linear von Kries adaptation model with the CMCCAT2000 primary matrix [14], which does a reasonable job of accounting for chromatic adaptation when shifting from one dominant illuminant to another [15]. The original white point primaries (R_w,G_w,B_w) are computed from the illuminant XYZ using the $\mathbf{M}_{\text{CMCCAT}}$ matrix, and the destination primaries (R_w',G_w',B_w') for D65 are computed using the same transform to be (0.9478,1.0334,1.0850).

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} R_w'/R_w & 0 & 0 \\ 0 & G_w'/G_w & 0 \\ 0 & 0 & B_w'/B_w \end{bmatrix} \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = \mathbf{M} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

$$\mathbf{M}_{CMCCAT} = \begin{bmatrix} 0.7982 & 0.3389 & -0.1371 \\ -0.5918 & 1.5512 & 0.0406 \\ 0.0008 & 0.0239 & 0.9753 \end{bmatrix} \quad (6)$$

The combined matrices for a white shift from standard illuminants B and A to D65 (whose chromaticities are given at the top of Table 1) and subsequent conversion from CIE XYZ to CCIR-709 RGB color space, are given in Eq.(7) as $\mathbf{C}_B$ and $\mathbf{C}_A$. Matrix $\mathbf{C}_{709}$ from Eq. (5) was concatenated with the matrix terms in Eq. (6) to arrive at these results, which may be substituted for $\mathbf{C}_{709}$ in Eq. (5) to get the adjusted RGB colors in the third row of Table 1 from the absolute XYZ values in the first row.

$$\mathbf{C}_B = \begin{bmatrix} 3.1273 & -1.6836 & -0.4867 \\ -0.9806 & 1.9476 & 0.0282 \\ 0.0605 & -0.2036 & 1.3404 \end{bmatrix}$$

$$\mathbf{C}_A = \begin{bmatrix} 2.9355 & -2.0416 & -0.5116 \\ -1.0247 & 2.1431 & -.0500 \\ 0.0732 & -0.1798 & 3.0895 \end{bmatrix} \quad (7)$$

Conventional CG Calculation

The standard approach in computer graphics color calculations is to assume all light sources are perfectly white and perform calculations in RGB color space. To display the results, a linear scale factor may be applied to bring the results into some reasonable range, and any values outside the sRGB gamut will be clamped.

We obtain an RGB value for the BlueFlower material from its published (x,y) chromaticity of (0.265,0.240) and reflectance of 24.3%. These published values correspond to viewing under standard illuminant C (simulated overcast), which is slightly bluer than D65. The linear RGB color for the flower material using the matrix $\mathbf{C}_{709}$ from Eq. (5) is (0.246,0.217,0.495), which differs from the D65 results in Table 1 by 10 ΔE* units using the CIE L*uv perceptual metric [16]. Most of this difference is due to the incorrect scene illuminant assumption, since the ΔE* between illuminant C and D65 is also around 10. This demonstrates the inherent sensitivity of color calculations to source color. Using the color corresponding to the correct illuminant is therefore very important.

The reason CG lighters usually treat sources as white is to avoid the whole white balancing issue. As evident from the third row in Table 1, careful accounting of the light source and chromatic adaptation is almost a no-op in the end. For white points close to the viewing condition of D65, the difference is small: a difference of just 1 ΔE* for illuminant B. However, tungsten is very far from daylight, and the ΔE* for illuminant A is more than 5, which is definitely visible. Clearly, if we include the source spectrum, we need to include chromatic adaptation in our tone-mapping. Otherwise, the differences will be very visible indeed -- a ΔE* of 22 for illuminant B and nearly 80 for illuminant A!

What if we include the source color, but use an RGB approximation instead of the full spectral rendering? Errors will creep in from the reduced spectral resolution, and their significance will depend on the source and reflectance spectra. Computing everything in CCIR-709 RGB for our BlueFlower example, the ΔE* from the correct result is 1 for illuminant B and nearly 8 for illuminant A. These errors are at least as large as ignoring the source color entirely, so there seems to be little benefit in this approach.

Relative Color Approximation

An improved method that works well for scenes with a single dominant illuminant is to compute the absolute RGB color of each material under the illuminant using a spectral precalculation from Eqs. (3) and (4). The source itself is modeled as pure white (Y,Y,Y) in the scene, and sources with a different color are modeled relative to this illuminant as (*Rs/Rw*,*Gs/Gw*,*Bs/Bw*), where (*Rw*,*Gw*,*Bw*) is the RGB value of the dominant illuminant and (*Rs*,*Gs*,*Bs*) is the color of the other source. In our example, the RGB color of the BlueFlower material under the three standard illuminants are those given in the second row of Table 1.

Prior to display, the von Kries chromatic adaptation in Eq. (5) is applied to the image pixels using the dominant source and display illuminants. The incremental cost of our approximation is therefore a single transform on top of the conventional CG rendering, and the error is zero by construction for direct reflection from a single source type. There may be errors associated with sources having different colors and multiple reflections, but these will be negligible in most scenes. Best of all, no software change is required – we need only precalculate the correct RGB values for our sources and surfaces, and the rest comes for free.

It is even possible to save the cost of the final von Kries transform by incorporating it into the precalculation,

computing adjusted rather than absolute RGB values for the materials, as in Eq. (7). We would prefer to keep this transform separate to preserve the colorimetric nature of the rendered image, but as a practical matter, it is often necessary to record a white-balanced image, anyway. As long as we record the scene white point in an image format that preserves the full gamut and dynamic range of our tristimulus pixels, we insure our ability to correctly display the rendering in any device's color space, now and in the future.

High Dynamic Range Images

Real scenes and physically-based renderings of real scenes do not generally fit within a conventional display's gamut using any reasonable exposure value (i.e., scale factor). If we compress or remap the colors to fit an sRGB or similar gamut, we lose the ability to later adjust the tone-scale or show off the image on a device with a larger gamut or wider dynamic range. What we need is a truly device-independent image representation, which doesn't take up too much space, and delivers superior image quality whatever the destination. Fortunately, such formats exist.

Since its inception in 1985, the *Radiance* physically-based renderer has employed a 32-bit/pixel RGBE (Red-Green-Blue-Exponent) format to store its high dynamic range output [17]. Predating *Radiance*, Bill Reeves of **Pixar** created a 33-bit log RGB format for the REYES rendering system, and this format has a public version contributed by Dan McCoy in 1996 to Sam Leffler's free TIFF library (www.libtiff.org). While working at **SGI**, the author added to the same TIFF library a LogLuv format that captures 5 orders of magnitude and the full visible gamut in 24 bits using a perceptual color encoding [18]. The 32-bit version of this format holds up to 38 orders of magnitude, and often results in smaller files due to run-length encoding [19]. Both LogLuv formats combine a logarithmic encoding of luminance with a linear encoding of CIE (u',v') chromaticity to cover the full visible gamut as opposed to the gamut of a specific device or medium.

Of the formats mentioned, only SGI's LogLuv TIFF encoding covers the full gamut and dynamic range of perceivable colors. The *Radiance* RGBE format spans a large dynamic range but is restricted to positive RGB values, so there are visible chromaticities it cannot represent. There is an XYZE version of the same format, but the associated quantization errors make it a poor choice. The Pixar 33-bit log format also has a restricted RGB gamut and only covers 3.8 orders of magnitude, which is marginal for human perception. Since the TIFF library is well tested and free, there is really no reason not to use LogLuv, and many rendering packages now output in this format. Even shareware browsers such as *ACDSee* are able to read and display LogLuv TIFF's.

Gamut Mapping

In order to fit a high dynamic range image into the limited color space of a conventional display, we need to apply one of the gamut compression techniques mentioned at the beginning of this section.

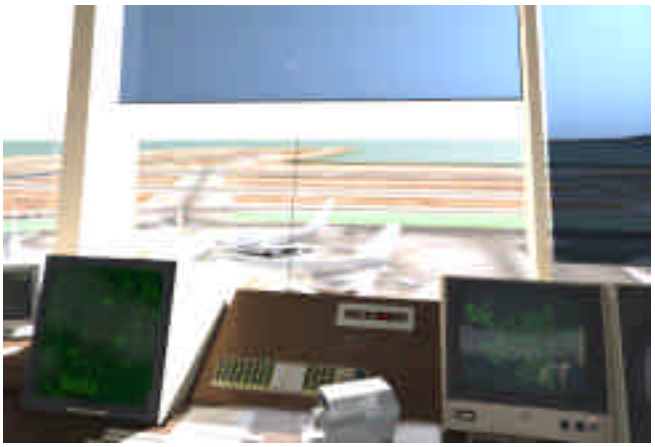

Figure 3. Radiance rendering of control tower clamped to limited display gamut and dynamic range.

Figure 4. The same rendering displayed using a visibility-preserving tone operator including glare effects.

Figure 5. A tone operator designed to optimize print contrast.

Specifically, we show how one might apply the third approach to display an image:

C. Scale colors on a curve determined by image content, as in a global histogram adjustment.

We assume that the rendering system has calculated the correct color at each pixel and stored the result in a high dynamic-range image format. Our task is then to examine this image and choose an appropriate mapping to our display. This is a difficult process to automate, and there is no guarantee we will achieve a satisfactory result in all cases. The best we can do is codify a specific set of goals and requirements and optimize our tone-mapping accordingly.

One possible goal of physically-based rendering is to assess visibility in some hypothetical environment or situation, or to recreate a situation that is no longer readily available (e.g., a plane crash). In such cases, we want to say that anything visible to an observer in the actual scene will be visible on the tone-mapped display. Conversely, if something is not visible on the display, we want to say that it would not be visible to an observer in the actual scene. This kind of visibility-matching operator was described in [20], and we show the result in Fig. 4. Fig. 3 shows the image mapped to an sRGB gamut using technique B to desaturate out-of-gamut colors. As we can see, some of the detail in the planes outside the window was lost to clamping, where it is preserved in the visibility-matching histogram-adjustment procedure in Fig. 4. An optional feature of our tone operator is the ability to simulate disability glare, which reduces visible contrast due to the harsh backlighting in the tower environment. This is visible as a slight haze in front of the monitors in Fig. 4.

Fig. 5 demonstrates another type of tone operator. This is also a histogram adjustment method, but instead of attempting to reproduce visibility, this operator seeks to optimize contrast over the entire image while keeping colors within the printable gamut. Especially in digital photo printers, saturated colors may be difficult to reproduce, so it may be desirable to darken an image to avoid desaturating some regions. We see that this method produces good contrast over most of the image.

Fig. 6 shows the global mapping of these three operators from world (rendered) luminance to display value (fraction of maximum). Where the naive linear operator clamps a lot of information off the top end, the two histogram adjustment operators present this information at a reduced contrast. This compression is necessary in order to bring out detail in the darker regions. We can see that the slopes match the linear operator near black in Fig. 7, deviating from the linear clamping operator above a certain level, where compression begins.

Fig. 8 plots the contrast optimizing tone operator against the world luminance distribution. Peaks in the luminance histogram correspond to increases in contrast, visible in the tone-mapping as a slight increase in slope. Since this is a log-log luminance plot, a small change in slope corresponds to a large change in contrast. The dip between 1.5 and 2.0 corresponds to a more gradual slope in the tone-mapping and lower contrast. In the low end, we see that this operator tends to provide more contrast to compensate for veiling reflection typical of glossy prints.

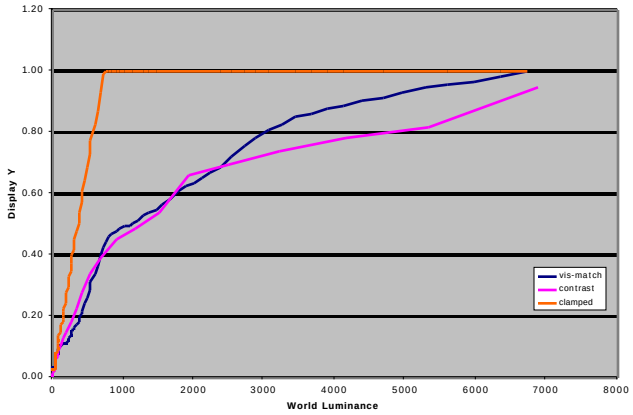

Figure 6. Comparison between three tone-mapping operators.

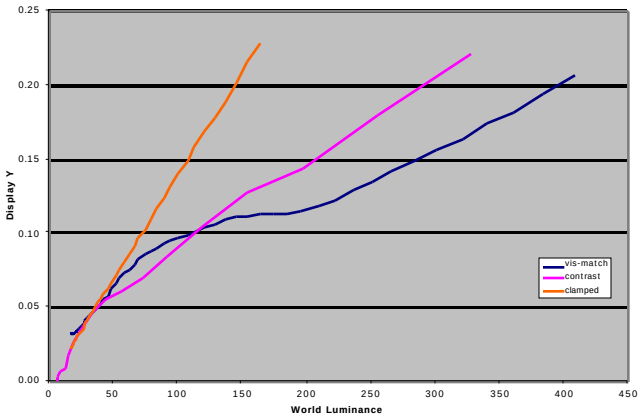

Figure 7. Close-up on darker region of tone-mappings.

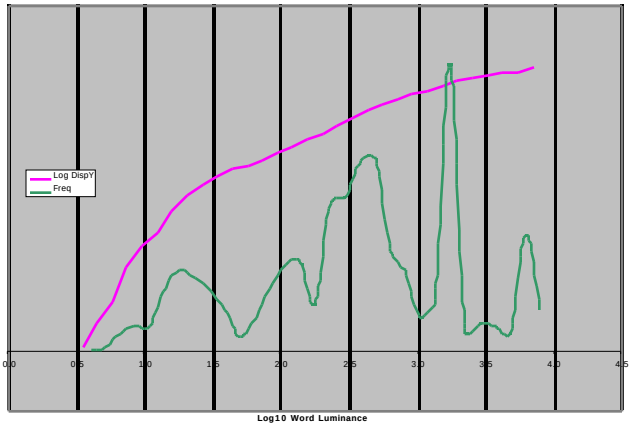

Figure 8. Good global tone operators produce greater contrast at peaks in the input histogram.

Conclusion

The recommendations we make in this paper for accurate color rendering may be summarized as follows:

1. Use a global illumination method with appropriate solutions for all of the phenomena being simulated.
2. Follow accurate spectral calculations with a good chromatic adaptation model to avoid color casts in the displayed image.
3. Substitute full spectral rendering with a relative color approximation for scenes with a single dominant illuminant.
4. Record images in a high dynamic range format to preserve display options (i.e., SGI LogLuv TIFF).
5. Base tone-mapping and gamut-mapping operators on specific goals, such as matching visibility or optimizing color or contrast.

Floating-point spectral calculations and high dynamic-range image manipulation are critical to accurate color rendering. The original approach of rendering directly in 24-bit RGB was recognized as hopeless and abandoned decades ago, but much of the mentality behind it remains with us today.

The methods outlined in this paper are not particularly expensive, neither in terms of implementation effort nor rendering cost. It's simply a matter of applying the right approximation. The author is not aware of any commercial software package that follows more than one or two of these principles, and it seems like a question of priorities.

Most of the money in rendering is spent by the entertainment industry, either in movies or in games. Little emphasis has been placed on accurate color rendering, but with the recent increase in mixed-reality rendering, this is beginning to change. Mixed-reality special effects and games require rendered imagery to blend seamlessly with film or live footage. Since reality follows physics and color science, rendering software will have to do likewise. Those of us whose livelihood depends on predictive rendering and accurate color stand to benefit from this shift.

References

1. Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, Ricardo Motta, A Standard Default Color Space for the Internet, www.w3.org/Graphics/Color/sRGB
2. Greg Ward, The RADIANCE Lighting Simulation and Rendering System, *Computer Graphics (Proceedings of SIGGRAPH 94)*, ACM, 1994.
3. Paul Debevec, Jitendra Malik, Recovering High Dynamic Range Radiance Maps from Photographs, *Computer Graphics (Proceedings of SIGGRAPH 97)*, ACM, 1997.
4. Alexander Wilkie, Robert Tobler, Werner Purgathofer, Combined Rendering of Polarization and Fluorescence Effects, *Proceedings of 12th Eurographics Workshop on Rendering*, June 2001.
5. Henrik Wann Jensen, Stephen Marschner, Marc Levoy, Pat Hanrahan, A Practical Model for Subsurface Light Transport, *Computer Graphics (Proceedings of SIGGRAPH 01)*, ACM, 2001.
6. Xiaodong He, Ken Torrance, François Sillion, Don Greenberg, A Comprehensive Physical Model for Light Reflection, *Computer Graphics (Proceedings of SIGGRAPH 91)*, ACM, 1991.
7. Jay Gondek, Gary Meyer, Jon Newman, Wavelength Dependent Reflectance Functions, *Computer Graphics (Proceedings of SIGGRAPH 94)*, ACM, 1994.
8. Holly Rushmeier, Ken Torrance, The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium, *Computer Graphics (Proceedings of SIGGRAPH 87)*, ACM, 1987.
9. Henrik Wann Jensen, Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps, *Computer Graphics (Proceedings of SIGGRAPH 98)*, ACM, 1998.
10. Jim Kajiya, The Rendering Equation, *Computer Graphics (Proceedings of SIGGRAPH 86)*, ACM, 1986.
11. Greg Ward Larson, Rob Shakespeare, *Rendering with Radiance*, Morgan Kaufmann Publishers, 1997.
12. Henrik Wann Jensen, *Realistic Image Synthesis Using Photon Mapping*, A.K. Peters Ltd., 2001.
13. Francois Sillion, Claude Puech, *Radiosity and Global Illumination*, Morgan Kaufmann Publishers, 1994.
14. C. Li, M.R. Luo, B. Rigg, Simplification of the CMCCAT97, *Proc. IS&T/SID 8th Color Imaging Conference*, November 2000.
15. Sabine Süsstrunk, Jack Holm, Graham Finlayson, Chromatic Adaptation Performance of Different RGB Sensors, *IS&T/SPIE Electronic Imaging*, SPIE Vol. 4300, January 2001.
16. Günter Wyszecki, W.S. Stiles, *Color Science*, J. Wiley, 1982.
17. Greg Ward, Real Pixels, *Graphics Gems II*, edited by James Arvo, Academic Press, 1992.
18. Greg Ward Larson, Overcoming Gamut and Dynamic Range Limitations in Digital Images, *IS&T/SID 6th Color Imaging Conference*, November 1998.
19. Greg Ward Larson, The LogLuv Encoding for Full Gamut, High Dynamic Range Images, *Journal of Graphics Tools*, 3(1):15-31 1998.
20. Greg Ward Larson, Holly Rushmeier, Christine Piatko, A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 4, December 1997.

Biography

Greg Ward (a.k.a. Greg Ward Larson) graduated in Physics from UC Berkeley in 1983 and earned a Master's in Computer Science from SF State University in 1985. Since 1985, he has worked in the field of light measurement, simulation, and rendering variously at the Berkeley National Lab, EPFL Switzerland, Silicon Graphics Inc., Shutterfly, and Exponent. He is author of the widely used *Radiance* package for lighting simulation and rendering.