



Spectral Primary Decomposition for Rendering with sRGB Reflectance

Ian Mallett¹  and Cem Yuksel¹ 

¹University of Utah

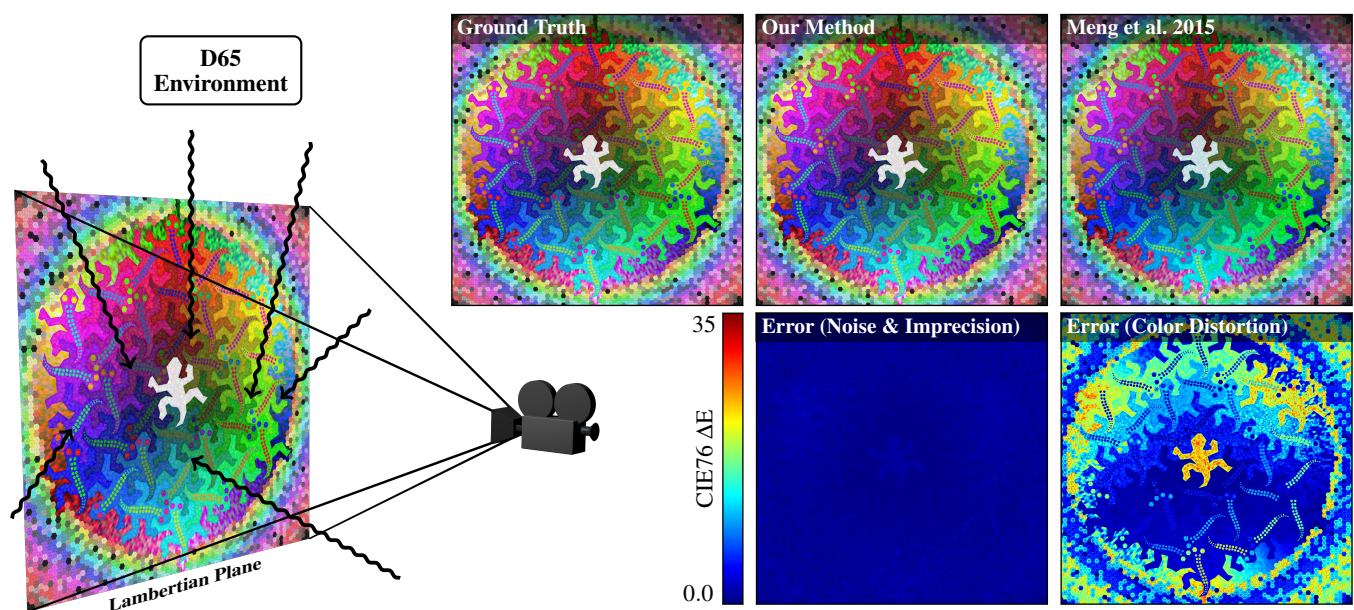


Figure 1: Spectral rendering of a texture containing the entire sRGB gamut as the Lambertian albedo for a plane under a D65 environment. In this configuration, ideally, rendered sRGB pixels should match the texture's values. Prior work by Meng et al. [MSHD15] produces noticeable color distortion, whereas our method produces no error beyond numerical precision and Monte Carlo sampling noise (the magnitude of the ΔE induced by this noise varies with the image because sRGB is perceptually nonlinear). Contemporary work [JH19] is also nearly able to achieve this, but at a significant implementation and memory cost.

Abstract

Spectral renderers, as-compared to RGB renderers, are able to simulate light transport that is closer to reality, capturing light behavior that is impossible to simulate with any three-primary decomposition. However, spectral rendering requires spectral scene data (e.g. textures and material properties), which is not widely available, severely limiting the practicality of spectral rendering. Unfortunately, producing a physically valid reflectance spectrum from a given sRGB triple has been a challenging problem, and indeed until very recently constructing a spectrum without colorimetric round-trip error was thought to be impossible. In this paper, we introduce a new procedure for efficiently generating a reflectance spectrum from any given sRGB input data. We show for the first time that it is possible to create any sRGB reflectance spectrum as a linear combination of three separate spectra, each directly corresponding to one of the BT.709 primaries. Our approach produces consistent results, such that the input sRGB value is perfectly reproduced by the corresponding reflectance spectrum under D65 illumination, bounded only by Monte Carlo and numerical error. We provide a complete implementation, including precomputed spectral bases, and discuss important optimizations and generalization to other RGB spaces.

CCS Concepts

• Computing methodologies → Reflectance modeling;

1. Introduction

Representing light using only three primaries (i.e. a *tristimulus* representation) is the standard in computer graphics. While cheaper than storing an entire visible spectrum, many colors cannot be represented—and many effects, particularly dispersion, cannot be correctly simulated. Even in simple scenes, without considerable attention when solving light transport, mixing spectral effects into this simplified representation leads to non-physical results, since tristimulus triples are not physical quantities.

Spectral renderers solve these problems by simulating the actual visible-light spectrum during light transport. For this to work, however, all scene assets must be accessible as spectra. Unfortunately, spectral data is rare, since existing content-authoring pipelines are primarily geared toward processing data using tristimulus representations (particularly in RGB spaces, especially BT.709's sRGB encoding), and most tools are completely incapable of working with spectral data. The typical solution to this problem is to generate (or sample from an implicit) full spectrum produced from a given tristimulus triple at render time. However, existing methods for this operation have been problematic, being computationally expensive, containing round-trip errors (the generated spectrum does not match the input tristimulus data), producing interpolation artifacts, and in the typical case, several of these at-once.

In this paper, we introduce a simple method that generates full reflection spectra directly from any given tristimulus input data. Each resulting spectrum satisfies the following crucial criteria:

- It is energy-conserving; i.e., the spectrum is within the range $[0, 1]$ for all wavelengths,
- It contains no round-trip error (up to numerical precision and Monte Carlo noise); the resulting spectrum matches the input tristimulus data,
- Smooth variation in the input triple results in smooth variation in the output spectrum,
- It varies relatively smoothly in wavelength, in accordance with the behavior of many natural materials, and
- The black and white points of the tristimulus color space produce perfectly flat spectra with all values being 0 or 1, respectively.

Our approach is simply to use a linear combination of three pre-computed *spectral primaries*, each corresponding to one of the three primary color components of the tristimulus representation. The consensus in all prior work was that such a *spectral basis* is impossible, but we constructively show that this conclusion was erroneous: in-fact, there are in fact infinitely many different sets of three spectral primaries.

This conclusion is applicable to the sRGB representation for input data, making it broadly useful for extant 3D assets. Our approach does not generalize well to wide-gamut spaces, largely because wider-gamut spaces contain colors that are not physically meaningful as reflectances. We discuss these issues in §7.

We also provide some details for achieving an efficient implementation in §5 and include full source code (including some example spectral primaries) of an example spectral renderer used to produce our results, in our supplementary materials.

2. Background

In the real world, light comes in different wavelengths. When looking at a light source, or object reflecting one, a (*radiometric*) *spectrum* of light enters our eyes. Under photopic (i.e., well-lit) viewing conditions, the typical response of the observer's eyes to this light is dominated by the amount that the spectrum stimulates *cone cells* of three types. Mathematically, this effectively takes an inner-product of the spectrum vector against three different basis vectors representing the wavelength sensitivities of the three types of cone cells.

The XYZ color space [CIE32] of the International Commission on Illumination (CIE) formalizes this process by defining three *standard observer functions*, $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$. The functions are linear transformations of the cone sensitivities, themselves inferred from perceptual experiments, and come in 2° and 10° observing variants. While the original CIE 1931 XYZ definitions are still widely used in graphics, there have been several improvements to the function definitions over the years. The current CIE 2006 XYZ standard [CIE06] is more perceptually accurate and is sampled at a higher resolution (refer to Figure 2). While we suggest using the 2006 10° functions, none of the results in this paper depend substantially on the variant chosen.

Given any spectrum \mathcal{S} , its CIE XYZ coordinates are computed with the vector Riemann sum:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \sum_{\lambda} \begin{bmatrix} \bar{x}(\lambda) \\ \bar{y}(\lambda) \\ \bar{z}(\lambda) \end{bmatrix} \mathcal{S}(\lambda). \quad (1)$$

The sRGB color space is the most-common color representation used in computer graphics. The white point $(1, 1, 1)$ of the sRGB space corresponds to CIE Standard Illuminant D65, a standardized *spectral power distribution* intended to be roughly representative of daylight. Idealized sRGB color values can be computed using the XYZ tristimulus values. First, a simple matrix multiplication is used to produce the RGB tristimulus (ℓ RGB, in linear space):

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = M^{-1} \left(\frac{1}{Y_{D65}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right), \quad (2)$$

where M is a constant 3×3 transformation matrix depending on D65 and the chromaticities defined in the BT.709 standard, and Y_{D65} is the Y-coordinate of D65. Then, a nonlinear transformation (i.e. *gamma*) and quantization is applied to convert ℓ RGB to sRGB. Clipping or tonemapping may be applied to bring the color into a displayable range, or to apply artistic direction.

Any two spectra that produce the same XYZ tristimulus values are called *metamers* and under the same *environment adaptation* they will appear as the same color, even if the spectra are different. Therefore, generating a visible-light spectrum from a given tristimulus color value is an underdetermined problem, with each tristimulus data point corresponding to infinitely many spectra. Additional constraints must be employed for deterministically selecting a particular spectrum for each tristimulus data point. A typical constraint is picking a spectrum that varies as little as possible [Smi99; MSHD15], motivated by the observation that many materials, especially natural ones, have smooth reflection spectra [Mal86].

3. Previous Work

Early work by MacAdam [Mac35] produced spectra with box shapes. This is not able to reproduce many colors and of-course does not result in smooth spectra, but it set the stage for modern work in computer graphics.

The pioneering work by Smits [Smi99] proposes formulating spectrum generation as an optimization problem. Energy-conservation is included as a form of soft-constraint due to the belief that energy-conservation might be impossible to satisfy (which is indeed the case for the downsampled RGB representation used). Like our approach, instead of solving the optimization problem separately for each input RGB color, which would be costly, linear combinations of a small number of precomputed spectra are used. However, the optimization formulation of Smits [Smi99] does not produce an acceptable solution using only three precomputed spectra for the RGB primaries, so they are combined with the spectra for CMYK primaries using a reasonable but nonetheless ad-hoc formulation. The results also contain considerable round-trip error.

Meng et al. [MSHD15] extend this optimization approach by adding more precomputed spectra, evaluated at the vertices of a grid placed along the 2D chromaticity xy plane of the CIE XYZ standard. This method precomputes and stores a relatively large number of spectra, one for each needed primary at each grid vertex. The in-between values are computed by interpolating the precomputed spectra, which leads to grid interpolation artifacts. The resulting spectra are then mapped to the solid of natural reflectances: a process that can introduce additional round-trip error. Very-saturated or -monochromatic colors are excluded to avoid numerical instabilities. As-compared to the method of Smits [Smi99], Meng et al. [MSHD15] can generate spectra for a much-wider range of colors, but they also state in their supplementary video that in their formulation not all valid sRGB colors within $[0, 1]^3$ form valid reflectances.

An alternative approach for generating a full spectrum from a given tristimulus value is using a large set of measured reflectance spectra and interpolating them for in-between points [WXS04]. This approach can be optimized by minimizing the necessary storage, applying principal component analysis (PCA) to reduce the dimensionality [OYH18]. The obvious advantage of this approach is that the spectra for the measured points within the tristimulus color space come from physical materials (as-opposed to an arbitrarily-chosen metamer, perhaps chosen to minimize some smoothness energy). On the other hand, the same argument does not hold for all other points that must use interpolated spectra. Indeed, interpolation using PCA can lead to non-physical spectra that are not energy-conserving. Moreover, covering the entire sRGB color space would require a large number of measured samples spanning the entire color space or extrapolation of spectra, which can quickly produce non-physical results. Furthermore, the variation of the resulting spectrum within the color space can include sharp changes and can even be discontinuous. This is not surprising, considering that each chosen measured spectrum is only a single metamer for the corresponding color value, and other metamers might exist in nature. Therefore, while using measured data for this problem is an interesting approach, further research is needed to address these important practical problems.

Very-recent work by Jakob and Hanika [JH19] creates a function space parametrized by coefficients. Input RGB data can be transformed via precomputation into these coefficients, and the result sampled as a spectrum at runtime. Significantly, this work was the first to upsample the entire sRGB gamut without introducing significant error (the precomputation relies on a table, leading to interpolation error, but any error introduced by this effect appears to be insignificant in the results they report). The authors note [Han19] that the coefficients have the same runtime memory footprint as an equivalent 32-bit floating-point RGB texture, but 10–16-bit precision is workable, and 8-bit has at-least been considered. However, for this reason, their technique is not directly comparable (especially in-terms of performance) to our work (we impose no particular limitation on bit-depth, but all results in this paper use 8-bit channels). Their work also produces smooth spectra and is able to convert wide-gamut RGB triples into valid reflectances in a controlled way.

We see our work as complementary to Jakob and Hanika [JH19]: our work requires no precomputation and has an even-simpler runtime model. We have only floating-point precision and the renderer's Monte Carlo error, whereas their work additionally has interpolation error (albeit this appears negligible). However, our spectra are less-smooth, and our approach cannot be used for wider-gamut spaces. Finally, we address the question of whether a spectral basis is possible, which is not the point of their work.

The optimization solution we present is similar to the work of Meng et al. [MSHD15] and especially Smits [Smi99], with subtle differences that lead to completely different conclusions about spectrum generation. The key difference is including D65 illumination within the optimization process. We show that, using our approach, linear combinations of only three precomputed spectral primaries, one for each primary color of the tristimulus representation, is sufficient for energy-conserving spectrum generation without round-trip errors (up to numerical precision) for the entire BT.709 color space. Moreover, we also show that such a set of spectral primaries is not unique and in-fact there exist infinitely many such sets.

4. Constructing Spectra

Data in the BT.709 color space is often used for defining properties of digital scene assets (commonly, in this space's sRGB encoding, for defining material properties, such as surface albedo). Since our goal is converting scene assets defined using sRGB colors to spectral data, we only consider reflection spectra. The BT.709 standard defines key photometric properties of the space, but it will be helpful to explicitly show how this translates to interpretation of sRGB scene data.

Like previous work, we begin by considering the rendered color of a surface under a *neutral illumination* condition. To be neutral, this must be (a scaling of) the radiance of the white point of the space (i.e. D65 for BT.709). For simplicity, we take the bidirectional reflectance distribution function (BRDF) of the surface material to be Lambertian (since the Lambertian BRDF has radiance independent of viewing angle, the following discussion is simplified), but other BRDFs work analogously. This setup is depicted in Figure 1.

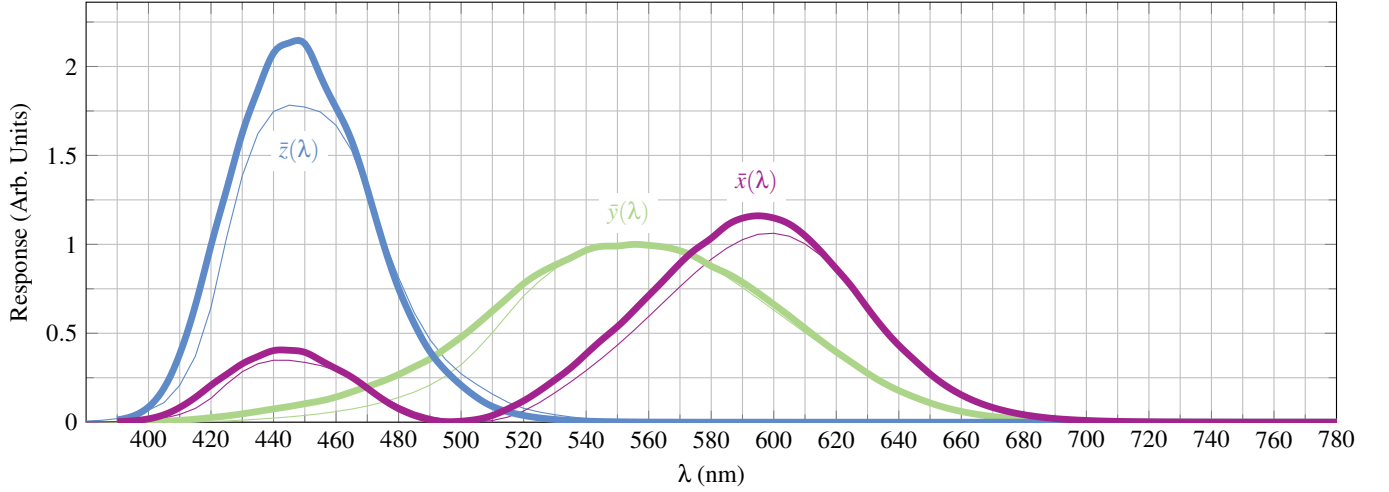


Figure 2: The CIE Standard Observer functions. **Thin lines:** 1931 standard, still widely used in graphics. **Thick lines:** newer 2006 standard, which makes improvements to cone response. See also the larger diagram in the supplemental materials.

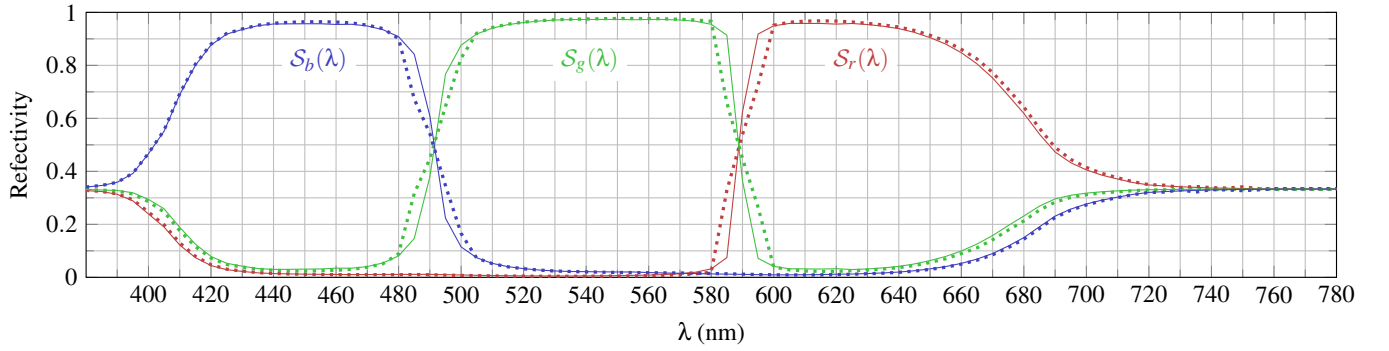


Figure 3: Two different sets of spectral primaries (**solid** and **dotted**), computed using different objective functions within the optimization. There are an infinite number of such sets, although they are all qualitatively similar.

By the definition of the BT.709 standard, the environment's D65 radiance *must* be rendered as the sRGB triple $\langle 1, 1, 1 \rangle$. Moreover, basic radiometry tells us that a white Lambertian has the same spectral radiance as a uniform environment—and therefore must be $\langle 1, 1, 1 \rangle$ also. Prior work on BT.709 spectral bases seems to have considered these facts incorrectly, typically by invoking (sometimes implicitly) illuminant E instead of D65, with the result that subsequent attempts to construct a feasible spectral basis failed. Using illuminant E does not correspond to the BT.709 color space in any way. In-fact, using it for the environment would actually result in a reddish color cast, instead of neutral illumination.

Mathematically, we let S represent the spectral reflectance of a given sRGB triple. In the setup above, the reflection off of a surface with this reflectance is $\mathbf{D}_{65} \odot S$, where \mathbf{D}_{65} is the discretized vector of D65 illumination and \odot represents the Hadamard product (i.e. element-wise multiplication). Using Equation 1, we can write the observed XYZ color of the reflected illumination as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \sum_{\lambda} \begin{bmatrix} \bar{x}(\lambda) \\ \bar{y}(\lambda) \\ \bar{z}(\lambda) \end{bmatrix} D_{65}(\lambda) S(\lambda). \quad (3)$$

Combining this with Equation 2, the equation for converting a spectrum S illuminated with D65 to the ℓ RGB triple it should appear on the screen as can be written:

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = M^{-1} \left(\frac{1}{Y_{D65}} \begin{bmatrix} (\mathbf{D}_{65} \odot \bar{\mathbf{x}})^T \\ (\mathbf{D}_{65} \odot \bar{\mathbf{y}})^T \\ (\mathbf{D}_{65} \odot \bar{\mathbf{z}})^T \end{bmatrix} S \right), \quad (4)$$

where $\bar{\mathbf{x}}$, $\bar{\mathbf{y}}$, and $\bar{\mathbf{z}}$ are the vectors representing the measured values of the CIE standard observer functions $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$. Rearranging the terms, we can write

$$\underbrace{\begin{bmatrix} (\mathbf{D}_{65} \odot \bar{\mathbf{x}})^T \\ (\mathbf{D}_{65} \odot \bar{\mathbf{y}})^T \\ (\mathbf{D}_{65} \odot \bar{\mathbf{z}})^T \end{bmatrix}}_{\mathbf{A} \in \mathbb{R}^{3 \times N}} S = \underbrace{Y_{D65} M}_{\mathbf{b} \in \mathbb{R}^{3 \times 1}} \begin{bmatrix} r \\ g \\ b \end{bmatrix}, \quad (5)$$

where N is the number of wavelengths in the discretized spectrum representation (e.g. $N = 81$ in the CIE 1931 XYZ standard). Thus,

we can pose the following constrained linear least-squares problem:

$$\begin{aligned} \text{minimize: } & \|AS - \mathbf{b}\| \\ \text{subject to: } & \mathbf{0} \leq S \leq \mathbf{1} \end{aligned} \quad (6)$$

for which a wide range of numerical solvers exist. It is not a surprising conclusion that there exists at least one valid *emission* spectrum corresponding to any color within the BT.709 space, since emitting metamers for these is in-fact how BT.709 displays work in the first place. What's somewhat more-surprising is that a valid *reflection* spectrum exists. Intuitively, since D65 is by-definition a metamer for the white point of a BT.709 display, and any individual primary reflects less than this, our approach “cuts out” the section of D65 that is “red”, “green”, and “blue”.

We have experimentally verified that for all discrete 24-bit sRGB color values, there is a valid solution that produces no round-trip error (up to numerical precision). In-fact, the minimization problem is not just solvable, but highly underdetermined (since $N \gg 3$), so there exist infinitely many metamers for each BT.709 color value that satisfy the energy-conservation constraint (the two exceptions being the white and black points, for which there are merely unique solutions, $S = \mathbf{1}$ and $S = \mathbf{0}$, respectively).

By itself, this is not a practical algorithm, since an optimization problem must be solved separately for each input RGB triple.

What is more interesting and a lot more useful in-practice is that, since the conversion in Equation 4 is linear, solving this optimization only for the three primaries of the tristimulus color space is sufficient to reproduce all other triples as a linear combination. We call this process *spectral primary decomposition*.

Let S_r , S_g , and S_b be the precomputed spectral primaries for the red, green, and blue primaries of the sRGB space, computed using the optimization process described above. The spectrum for any given ℓ RGB triple (r, g, b) can then be computed using a simple linear combination of the spectral primaries:

$$S = r S_r + g S_g + b S_b. \quad (7)$$

Note that each one of S_r , S_g , and S_b can be selected from infinitely many metamers. However, simply picking any metamer for S_r , S_g , and S_b would not guarantee that the resulting spectrum S would be energy-conserving. To enforce energy-conservation, while solving the optimization problem, we must introduce an additional constraint—that the spectral primaries form a partition of unity:

$$S_r(\lambda) + S_g(\lambda) + S_b(\lambda) = 1 \quad (8)$$

for all wavelengths λ . This constraint ties together the metamer selections for the three primaries, and it is an obvious constraint, considering that the white point $\langle 1, 1, 1 \rangle$ must form $S = \mathbf{1}$.

Even with this additional constraint, the optimization problem of Equation 6 is still underdetermined. Therefore, there are infinitely many choices for a valid spectral basis. In our experiments, we arbitrarily pick spectral primaries that minimize their maximum values or the maximum differences between consecutive wavelengths, shown in Figure 3, but any other sensible metric could be used instead. We leave further arguments about the “right” energy function to minimize to future work. The curves for the resulting spectral

primaries are somewhat different, but both form energy-conserving spectra with no round-trip error (up to numerical precision). Note that this also implies that they are perceptually identical on the first illumination bounce of D65.

We note that these conclusions constructively disprove many statements in various prior works that claimed that some sRGB values do not represent valid reflectances, and more-broadly that such a spectral basis was impossible.

5. Implementation Improvements

A naïve way of implementing a spectral renderer would be assigning a wavelength for each ray and accumulating a discretized spectrum for each pixel. Many spectral renderers still do basically this, leading to the widespread presumption that spectral rendering is necessarily many factors slower than tristimulus rendering. We discuss several key improvements that the implementer should consider, to dispel this notion:

The hero-wavelength sampling [WND*14] approach assigns multiple wavelengths to each ray with constant wavelength separation, thereby significantly improving the rendering performance, since with vectorization multiple wavelengths execute for a similar cost as one. The typical vector width on modern CPUs is 8, but even with 4, such as on GPUs, with hero-wavelength sampling, color typically converges far-faster than other sources of variance.

A refinement built upon hero-wavelength sampling is to observe that spectral data—such as in-particular our basis of spectral primaries—can be interleaved so that the data for all wavelengths in the hero sample can be loaded in a single vector memory access, instead of multiple scalar accesses spaced over the spectrum. In-addition to reducing the number of loads, this reduces shuffling within registers and significantly improves caching performance owing to the improved locality.

If the goal of spectral rendering is generating an image for tristimulus display (as is nearly-always the case), storing a full spectrum per-pixel is not actually necessary. Instead, the traversed wavelengths can be converted per-sample to CIE XYZ before being reconstructed into the framebuffer. This works because the inner-product is mathematically linear. This approach does invoke the spectrum-to-XYZ conversion operation per-sample, but this is a minor cost—with the above interleaving, for example, the entire conversion for a hero-wavelength sample is just three dot-products (typically one assembly instruction each), with the main cost coming from memory access to the operands. In-return, the storage requirements for the framebuffer are reduced dramatically, and all quantization problems related to discretization of the spectrum within the framebuffer are eliminated. We would like to see this improvement implemented more-widely than it seems to be.

6. Results

Results in this paper were rendered using a demonstration spectral path tracer, which we release in its entirety as part of our supplemental materials. The renderer is designed primarily to be easy to understand (and so lacks an acceleration structure and explicit vectorization).

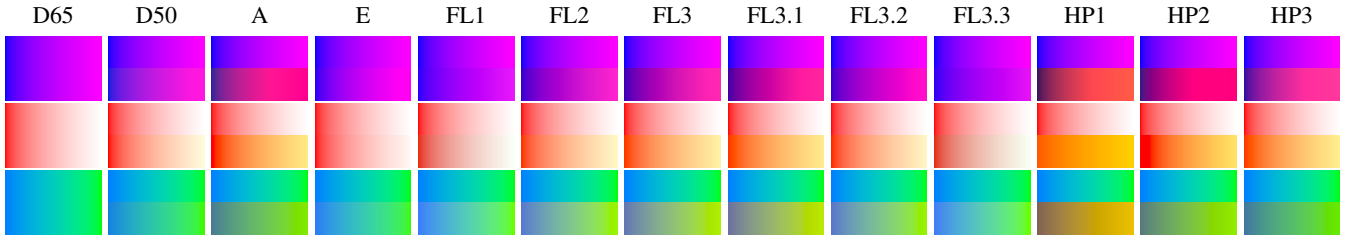


Figure 4: Three example gradients (*each row, top*) and their renderings (*each row, bottom*) under different CIE illuminants (*columns*). As-expected, the result matches exactly for D65, and in all cases our output remains smooth. Stability is mathematically guaranteed for any illuminant and any smooth gradient (refer to text). More illuminants and gradients can also be tested using the provided supplemental code.

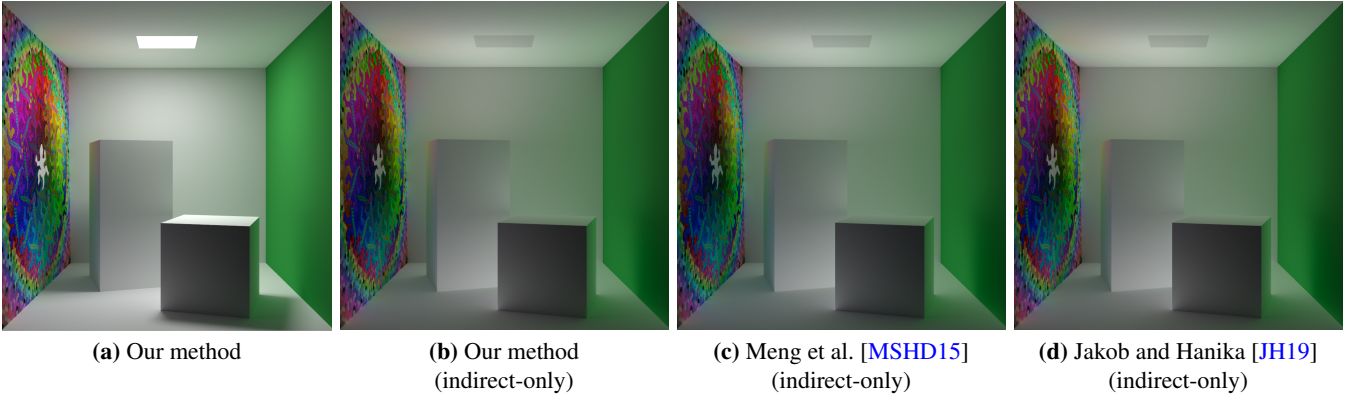


Figure 5: CORNELL BOX with an sRGB texture (left wall), the original dataset’s real-world measured spectrum (right wall), and simple procedural spectra (other surfaces). All images were rendered using the CIE 1931 observer functions.

Spectral primary decomposition is stable under different illuminants in the sense that smoothly varying input colors (i.e. gradients) result in smoothly varying output, which is desirable for intuitive results. In prior work, this feature must be demonstrated experimentally, but we can also prove it mathematically: since every spectrum generated by our algorithm is a linear combination of three spectra, any smooth variation in the input ℓ RGB triple must result in a smooth variation in the output spectrum. Moreover, again because of linearity, the output spectrum can be viewed as a smooth weighting of the three basis spectra pre-multiplied by the illuminant. Therefore, the output must also be well-behaved, regardless of the illuminant. Following the example of Jakob and Hanika [JH19], we show a few gradients and several CIE illuminants to demonstrate this in Figure 4 (each gradient is linear in ℓ RGB, each illuminant is normalized by its Y-coordinate, and results are clipped, as-necessary, for display). Arbitrary illuminants and gradients can be tested using the provided supplemental code.

In our supplemental material, we also include a self-contained interactive HTML+JavaScript file that shows precomputed spectral primaries and allows performing round-trip experiments for the entire BT.709 color space. Both sets of spectral primaries shown in Figure 3 are included. It can be seen using this file that our method produces minimal round-trip error bounded by numerical precision with both sets of spectral primaries. We also provide a third set of spectral primaries generated using a flat spectrum for the optimization, instead of the correct D65 illumination. This is analogous to

the optimization approaches used in prior work [Smi99; MSHD15]. As expected, this leads to the substantial round-trip error which has motivated the use of more-complicated spectrum generation methods in prior work.

Figure 1 shows a comparison of our method to Meng et al. [MSHD15], where a planar quad with an sRGB texture, containing all possible 24-bit color values of the sRGB color space, is illuminated with uniform D65 environment illumination, rendered using a spectral renderer with a large number of Monte Carlo samples. Under these conditions, ideally, the resulting image should match the input texture. Notice that our method closely matches the input texture values with minimal error (which is due to Monte Carlo sampling noise and numerical-precision errors). The perceptual magnitude of this noise varies slightly based on the texture color because BT.709 is not perceptually linear. The results are generated using the solid-lined primaries shown in Figure 3, though the other set of spectral primaries produces virtually identical results. Indeed, in-theory, any spectral basis should produce the same result, since this scene only contains one bounce of illumination.

A deterministic round-trip experiment produces the same outcome as this figure, but without Monte Carlo noise. The absolute maximum error (due to numerical precision) with our method is bounded by 1.851×10^{-5} . No particular consideration of numerically robust algorithms was performed, but this is already two orders of magnitude below the discretization error of sRGB.

In-comparison, the method of Meng et al. [MSHD15] produces substantial color shifts over a large portion of the texture. This is expected, since Meng et al. [MSHD15] perform the optimization only for the 2D chromaticity plane, and colors that are away from this plane can incur a significant amount of error. Since they do not use D65 illumination in their optimization process, the resulting spectra produces low error only near the precomputed sample locations near the 2D chromaticity plane. The method of [JH19] achieves similar quality to our results when 32-bit floating-point coefficients are used. However, this is not really analogous to our method, which uses only 8-bit textures at runtime.

Figure 5 shows a simple Cornell box scene containing a wall with sRGB texture, a green wall with real-world measured reflection spectrum, and procedurally-generated spectra for other materials, rendered using the provided spectral pathtracer. We explicitly visualize the indirect illumination to demonstrate that our algorithm remains well-behaved after multiple bounces. Like Jakob and Hanika [JH19], our algorithm produces stable, energy-conserving results, without unexpected color shifts. This is actually a special case of our above discussion of stability. Meng et al. [MSHD15] introduces color shifts into the primary illumination (refer to Figure 1), and so the shifts are also present in the indirect illumination.

7. Discussion

The majority of 3D assets are authored in sRGB, for which our approach is designed. However, as the industry moves toward wider-gamut spaces, one might consider extending our work to alternate RGB spaces.

This turns out to be broadly impossible. In-addition to the BT.709 primaries used by sRGB and discussed throughout, we tried BT.2020, DCI-P3 D65, Adobe RGB, NTSC RGB, PAL RGB, ProPhoto RGB, and Adobe Wide-Gamut RGB—every one of which encompasses a larger (albeit not-always-enveloping) set of $L^*a^*b^*$ colors than BT.709. Of these, only PAL RGB produces a feasible basis, which is relatively unsurprising since it is very similar to BT.709, whereas the others are significantly larger.

The core problem is that, unlike for BT.709, colors in these spaces do not necessarily represent valid reflectances.

Generating a physically-valid reflectance spectrum for a physically-impossible reflectance should be considered a philosophical issue that necessarily involves the content-authoring pipeline. The typical presumption, which has been applied by prior work for BT.709 as-well, seems to be that renderers should attempt to correct the issue at render-time. This allows for wide-gamut data to be used unmodified, but has the major disadvantage that colors change when rendered. By projecting the colors into the feasible region in a principled way, as do Jakob and Hanika [JH19], the shift can be made somewhat predictable—although of-course it still happens. Another approach might be to have artist tools clamp material data to physically-plausible reflectances in the first place, during the modeling process. This would likely make modeling non-intuitive, and would pose problems when chaining wide-gamut tools not explicitly created for the 3D modeling pipeline. Most-interestingly, perhaps a new color space could be constructed explicitly for the

purpose of modeling feasible reflectances. Finally, correct results can always be obtained by using real spectral data.

We note that the fact that some chromaticities in wide-gamut spaces do not represent reflectances is completely separate from the use of wide-gamut display devices. Any spectral renderer is capable of producing results for a wide-gamut display device, simply by using the appropriate XYZ-to-RGB conversion matrix.

8. Conclusion

We have presented a new method for generating physically valid spectral reflectances from input sRGB triples. Our approach is simply a linear combination of three spectra corresponding to the BT.709 primaries—a spectral basis. We have shown that by correctly considering the space’s white point (i.e. D65 illumination) in the optimization, the problem of generating these three spectral primaries becomes feasible. We provide full sample code and pre-computed data for our approach and experiments to aid adoption in our supplementary materials.

Acknowledgments

We wish to thank “Ita” for permission to use the lizard texture, Vidhi Zala for suggesting using MATLAB’s numerical solvers instead of SciPy’s, and authors of previous work for clarifications and making implementations both available and easy-to-use.

References

- [CIE06] CIE. *Fundamental chromaticity diagram with physiological axes*. Tech. rep. Vienna, Austria, 2006 2.
- [CIE32] CIE. “Commission internationale de l’éclairage proceedings, 1931”. *Cambridge University Press Cambridge* (1932) 2.
- [Han19] HANIKA, JOHANNES. personal communication. June 17, 2019 3.
- [JH19] JAKOB, WENZEL and HANIKA, JOHANNES. “A Low-Dimensional Function Space for Efficient Spectral Upsampling”. *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, 147–155 1, 3, 6, 7.
- [Mac35] MACADAM, DAVID L. “Maximum visual efficiency of colored materials”. *JOSA* 25.11 (1935), 361–367 3.
- [Mal86] MALONEY, L. T. “Evaluation of linear models of surface spectral reflectance with small numbers of parameters”. *Journal of the Optical Society of America* 3.10 (1986), 1673–1683 2.
- [MSHD15] MENG, JOHANNES, SIMON, FLORIAN, HANIKA, JOHANNES, and DACHSBACHER, CARSTEN. “Physically Meaningful Rendering using Tristimulus Colours”. *Computer Graphics Forum* 34.4 (2015), 31–40. DOI: [10.1111/cgf.12676](https://doi.org/10.1111/cgf.12676) 1–3, 6, 7.
- [OYH18] OTSU, H., YAMAMOTO, M., and HACHISUKA, T. “Reproducing Spectral Reflectances From Tristimulus Colours”. *Computer Graphics Forum* 37.6 (2018), 370–381. DOI: [10.1111/cgf.13332](https://doi.org/10.1111/cgf.13332) 3.
- [Smi99] SMITS, BRIAN. “An RGB-to-spectrum conversion for reflectances”. *Journal of Graphics Tools* 4.4 (1999), 11–22. DOI: [10.1080/10867651.1999.10487511](https://doi.org/10.1080/10867651.1999.10487511) 2, 3, 6.
- [WND*14] WILKIE, A., NAWAZ, S., DROSKE, M., et al. “Hero Wavelength Spectral Sampling”. *Proceedings of the 25th Eurographics Symposium on Rendering*. EGSR ’14. Lyon, France: Eurographics Association, 2014, 123–131. DOI: [10.1111/cgf.12419](https://doi.org/10.1111/cgf.12419). URL: <http://dx.doi.org/10.1111/cgf.12419> 5.
- [WXS04] WANG, QIQI, XU, HAIYING, and SUN, YINLONG. “Practical construction of reflectances for spectral rendering”. (2004). URL: <https://dspace5.zcu.cz/handle/11025/9733> 3.