EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

A Low-Dimensional Function Space for Efficient Spectral Upsampling

Wenzel Jakob[1] and Johannes Hanika[2,3]

[1]EPFL [2]KIT [3]Weta Digital

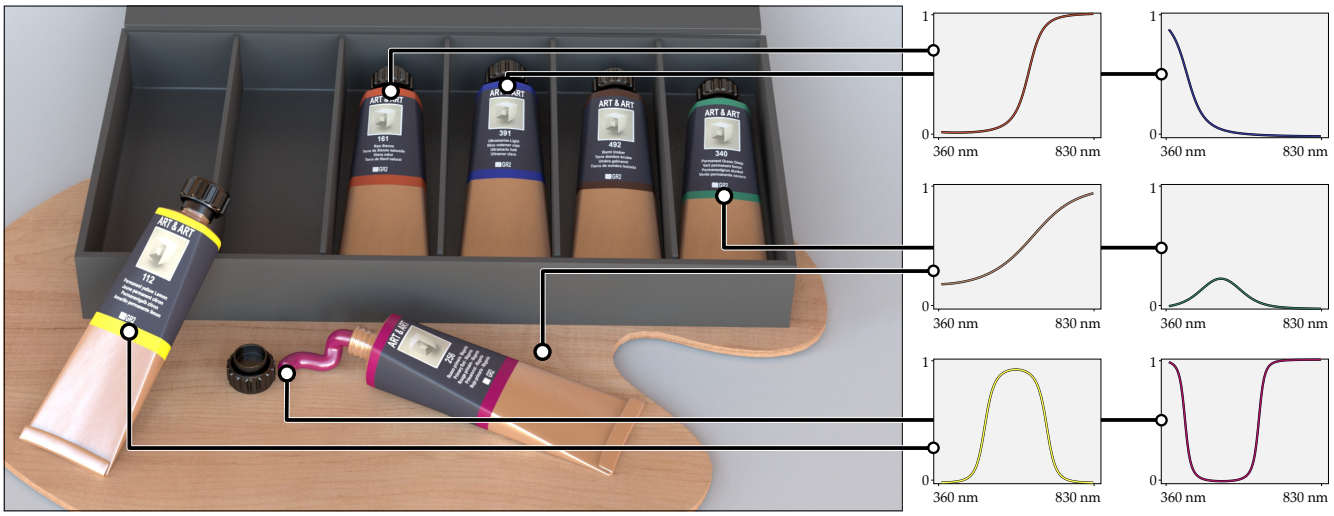


Figure 1: ***Left.*** *A spectral rendering performed using the proposed technique. This scene uses a variety of RGB textures that have been converted into reflectance spectra.* ***Right.*** *Plots of highlighted surface regions over the visible range.*

Abstract

We present a versatile technique to convert textures with tristimulus colors into the spectral domain, allowing such content to be used in modern rendering systems. Our method is based on the observation that suitable reflectance spectra can be represented using a low-dimensional parametric model that is intrinsically smooth and energy-conserving, which leads to significant simplifications compared to prior work. The resulting spectral textures are compact and efficient: storage requirements are identical to standard RGB textures, and as few as six floating point instructions are required to evaluate them at any wavelength. Our model is the first spectral upsampling method to achieve zero error on the full sRGB gamut. The technique also supports large-gamut color spaces, and can be vectorized effectively for use in rendering systems that handle many wavelengths at once.

CCS Concepts

• ***Computing methodologies*** → ***Reflectance modeling;***

1. Introduction

Physically-based rendering techniques are able to generate images of striking realism using a detailed simulation of the interaction of light and matter. In the pursuit of realism, many modern rendering systems have recently started to perform the underlying light transport simulation in the spectral domain, usually covering the visible range with wavelengths from roughly 360 to 830 nanometers.

This trend is motivated by several benefits of spectral color representations: spectral quantities faithfully describe the physical process that gives rise to color, which simplifies interfacing rendering systems with measured data or parametric models derived from first principles. Many modern reflectance models have a natural dependence on wavelength, e.g. to account for iridescence in thin layers [IA99, BB17] or diffraction from surface microstructure [DWMG15, WVJH17, YHW*18]. Products of reflectance spectra, which are needed e.g. to simulate multiple scattering, have been found to be in better agreement with real-world reflectance compared to products of RGB triplets [Pee93, WEV02, FHF*17]. Another benefit of a higher-dimensional representation is the ability to model metamerism.

On the flipside, spectral color representations introduce additional complexities: first, a numerical integration over wavelengths is required, which can introduce chromatic noise in a Monte Carlo renderer. This is typically ameliorated by using some form of vectorization to integrate over multiple wavelengths at once. A more severe issue is that scene specifications will generally only provide spectral data for a small subset of components such as materials, sensors, and light sources. A practical rendering system must thus be able to deal with "legacy" content based on tristimulus representations such as RGB. This article proposes a versatile solution to the latter problem, specifically the conversion of tristimulus reflectance data (e.g. textures) into an equivalent spectral representation.

Reflectance spectra due to reflection from absorbing surfaces are typically very smooth functions in the visible range[†]. Although reflectance spectra exist in a wide variety of shapes, we tend to observe functions that are well-approximated by constant (white, black), approximately linear, or peaked curves with one (green, yellow) or two modes (blueish-purple). Figure 2 shows a few examples from a color checker.

Spectral upsampling is a highly ill-posed problem, since each tristimulus color corresponds to an infinite dimensional subspace of potential reflectance spectra. There is thus a large design space of admissible solutions, making it feasible to impose a number of additional desiderata:

1. The composition of mappings from tristimulus values to spectra and the reverse should generally be an identity, i.e.

   $$\mathrm{rgb}(\mathrm{spec}(\mathbf{b})) = \mathbf{b}.$$

   This is not always possible, e.g. for very saturated and imaginary colors, in which case we want to minimize

   $$\|\mathrm{rgb}(\mathrm{spec}(\mathbf{b})) - \mathbf{b}\| \approx 0,$$

   where $\|\cdot\|$ is an arbitrary perceptual metric that defines the gamut mapping strategy being used.

2. Spectra should be as smooth as possible and furthermore have a smooth dependence on the input tristimulus value.

[†] In the UV and IR, absorption bands tend to cause sharp transitions. Also, wave-optical phenomena such as iridescence produce more complex oscillatory spectra and are not considered in this article. Our method also does not apply to light source spectra, which are often highly discontinuous due to stimulated emission.

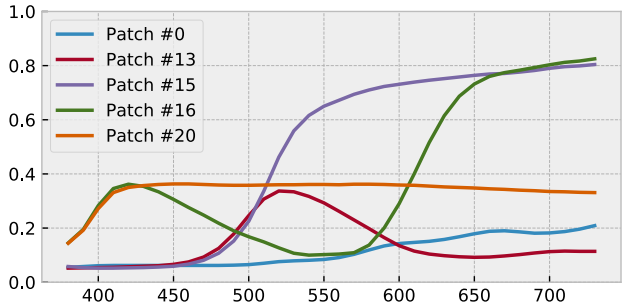


Figure 2: *Spectral measurements of several X-rite color checker patches in the 380 to 730 nm range.*

3. Evaluation must be extremely efficient, as it will occur billions of times during a typical rendering.
4. The code that implements evaluation of the spectrum should be compact enough so that it can be inlined into shader code.
5. Memory latency and throughput are key limiting factors in the performance of modern rendering systems, hence it is crucial that the conversion does not negatively affect them.
6. Most spectral rendering systems use SIMD instruction sets such as SSE or AVX to integrate over multiple wavelengths at once. The conversion must thus be amenable to vectorization.

Our method is based on the observation that a simple analytic model can satisfy all of the above criteria. It produces smooth spectra that are intrinsically energy-conserving, which leads to significant simplifications compared to prior work that has focused on this problem. Numerical evaluation consists of two steps: a brief pre-processing step at scene load time translates RGB texel values into parameters of an analytic model, which is then evaluated at render time. Only three coefficients are needed to parameterize the spectrum, which means that storage requirements of transformed textures are identical to those of ordinary RGB textures, and no additional memory traffic occurs at render time. The evaluation step is trivially vectorizable and extremely efficient, requiring as few as six floating point operations. Our model achieves low color reproduction errors on a variety of color spaces, and zero error on the commonly used sRGB gamut. A reference implementation of all components is provided in the supplemental material.

2. Previous Work

An important constraint that distinguishes *reflectance spectra* from *emission spectra* is that they conserve energy and thus cannot exceed a value of one at any wavelength. This restriction is subtly related to color saturation: a color appears more saturated when the spectrum contains narrow peaks, i.e. when the ratio between values at different wavelengths is maximized. Narrowing a peak while maintaining its maximum, however, will make the color darker since brightness is related to the integral of the spectrum. This has been recognised early on [Sch19], and the corresponding *gamut of valid reflectances* was constructed sixteen years later [Mac35b]. Note that we only target reflectance spectra in this article.

The first spectral upsampling methods can be found in the literature from many decades ago. Due to the growing importance of realistic color and spectral transport, a number of increasingly sophisticated methods have been proposed since then.

MacAdam. The technique of MacAdam [Mac35a] is mainly interesting for its theoretical merit. The tristimulus to spectrum upsampling method is a byproduct of a constructive proof of theoretical limits to the brightness of colors of certain saturation. The spectra are box (or inverted box) shaped and only consist of a rising and a falling edge. They are thus always able to represent colors of maximum brightness for any given saturation, but are not a good match for natural spectra that are usually smooth (cf. Figure 2).

Smits. The technique of Smits [Smi99] is also based on a box basis that is split into ten discrete bins. The bin values are derived using

a snake-based optimisation that accounts for energy conservation as well as the target RGB color of the spectrum. While this works well inside the sRGB gamut, the approach can become unstable for finer resolutions (more than ten spectral bins) and wider gamuts (for instance Rec. 2020). This is partly because arbitrarily saturated color and high brightness cannot be achieved in general [Mac35b].

Meng et al. The method of Meng et al. [MSHD15] is closest to our goals, hence we will discuss it in greater detail. The method supports wide gamut color spaces and produces smooth output spectra, but it is characterized by several undesirable aspects:

1. **Energy conservation.** Meng et al. optimize the smoothness of their spectra without constraining them to be physical (i.e. bounded by 1). The spectra must later be re-scaled, which introduces considerable errors even on "simple" color spaces such as sRGB. In contrast, our optimization minimizes colorimetric error on a function space of intrinsically energy-conserving spectra. Our solutions are generally very smooth but can also encode sharp or rectangular peaks when this is needed to satisfy the optimization objective, enabling us to reproduce the full sRGB gamut with zero error.
2. **Dimension.** Meng et al. precompute spectra for 2D *xy* chromaticity coordinates, while our method uses a 3D tabulation that also accounts for brightness. This is based on the observation that spectra near the gamut's rim of maximum brightness must necessarily approximate MacAdam's box-shaped spectra, while spectra of lower brightness can be smoother. There should thus be different shapes of spectra for different levels of brightness.
3. **Storage.** Meng et al. rely on a table of dense spectral discretizations ($\sim$ 60.7 KB, i.e. about the size of the L1 cache of modern processors) that will be accessed at least 4 times as part of every texture lookup. Since the stored functions are very smooth, a dense discretization seems excessive. Our method also relies on a tabulation that is even larger—however, a crucial difference is that it must only be accessed once when the texture is loaded from disk, and we also do not discretize the spectra themselves.
4. **Meshing.** Meng et al. mesh the spectral locus using quadrilateral interior cells and triangular boundary cells. This approach is very general and works for any RGB color space that lies within the spectral locus, but lookups on such an irregular cannot be vectorized effectively and thus lead to comparatively slow performance. We instead target specific RGB color spaces, which allows for a much simpler and more regular discretization. The supplemental material includes optimization code and precomputed parameters for four color spaces: sRGB (Rec. 709), Rec. 2020, ProPhoto RGB, and ACES 2065-1, which is widely used by the film industry.

Otsu et al. The most recent spectral upsampling technique by Otsu et al. [OYH18] employs a clustered principal component analysis (PCA) to reconstruct spectra that resemble natural reflectance spectra from a database. The method organizes *xy* chromaticity space into a kd tree containing eight clusters and a PCA per cluster to derive spectra from *XYZ* values. The tables used here are a bit smaller than those of Meng et al., but still relatively large (4.875 KB at the low 10nm bin resolution used in the paper.)

There are three issues with this approach: nearby colors that cross cluster boundaries can produce very different spectra, since there is no interpolation across clusters. This leads to color discontinuities in renderings. Secondly, output spectra often exceed the $[0,1]$ range including significant negative regions and must be clamped, which results in color reproduction errors. Finally, the dependence on brightness is affine, hence the method cannot alter the peakedness of spectra based on this parameter.

3. Method

The general problem that we wish to solve is to find a smooth spectrum $\hat{f}(\lambda)$ that maps to a RGB color $\mathbf{b} \in [0,1]^3$ in a specified color space. If an exact match is impossible, the spectrum should approximate the input color as closely as possible. We formalize this objective as the following optimization problem:

$$\hat{f} = \arg\min_f \left\| \mathbf{b} - \mathbf{T} \int_\Lambda f(\lambda)\, W(\lambda)\, \mathbf{xyz}(\lambda)\, \mathrm{d}\lambda \right\|, \tag{1}$$

where W represents the spectral power distribution of the white point (e.g. D65), $\mathbf{xyz}(\lambda)$ denotes the CIE 1931 color matching functions in vectorial form, $\mathbf{T} \in \mathbb{R}^{3\times 3}$ is a color transformation matrix that maps from CIE XYZ values to the RGB color space,[‡] and the integration domain $\Lambda = [360, 830]$ covers the visible spectrum. We define the norm $\|\cdot\|$ as CIE76 ΔE to quantify the perceptual magnitude of color differences. Finally, we use a composite Simpson 3/8 rule to compute the above integral with discretization of $\sim$1.6nm (3$\times$ the resolution of the CIE color matching functions, which we interpolate linearly.)

Instead of solving for an arbitrary discretized function such as a piecewise constant or linear interpolant [MSHD15], we found that a simple analytic model can be sufficiently expressive to yield high-quality solutions to the above optimization. We specifically set

$$f(\lambda) = S(c_0\lambda^2 + c_1\lambda + c_2) \tag{2}$$

where c_i are coefficients of a second-order polynomial and S is a sigmoid function that maps the interval $(-\infty, \infty)$ to $[0,1]$. The sigmoid function encodes the energy conservation constraint: due to the nonlinear mapping, nonphysical spectra with values outside of the interval $[0,1]$ cannot be produced irrespective of the model parameters.

Note that a number of functions with a sigmoidal shape exist, such as the arc tangent, the hyperbolic function, and the logistic function. All of them are in principle admissible, but we prefer a simple algebraic variant that does not require the evaluation of transcendental functions:

$$S(x) = \frac{1}{2} + \frac{x}{2\sqrt{1+x^2}}. \tag{3}$$

The combination of parabola and nonlinearity allows the model

‡ This formulation is specific to the human visual system, since it optimizes projected errors on the subspace spanned by the CIE color matching functions. Targeting other subspaces (e.g. for a camera), requires a different set of response curves—changing the linear transform $\mathbf{T}$ is not enough.

Algorithm 1 Evaluation of the spectral model $f(\lambda)$

```
1: function f(c0, c1, c2, wl)
2:     # Evaluate polynomial for wavelength 'wl'
3:     x = fma(fma(c0, wl, c1), wl, c2)
4:
5:     # Evaluate nonlinear map
6:     return fma(.5 * x, rsqrt(fma(x, x, 1)), .5)
```

to produce a number of shapes that are commonly seen in reflectance spectra, including a single mode (e.g. green spectra), two separated peaks (e.g. purple spectra), flat regions (maximally bright colors), and constant spectra (e.g. white and black). A number of examples are shown in Figure 1.

Evaluating the composition of the polynomial and sigmoid (Algorithm 1) requires as little as six floating point operations when implemented using fused multiply-additions ("`fma`") and reciprocal square root operations ("`rsqrt`") provided by current processor instruction sets. The evaluation is trivially vectorizable on modern instruction sets such as SSE, AVX and AVX512. We provide reference implementations in the supplemental material.

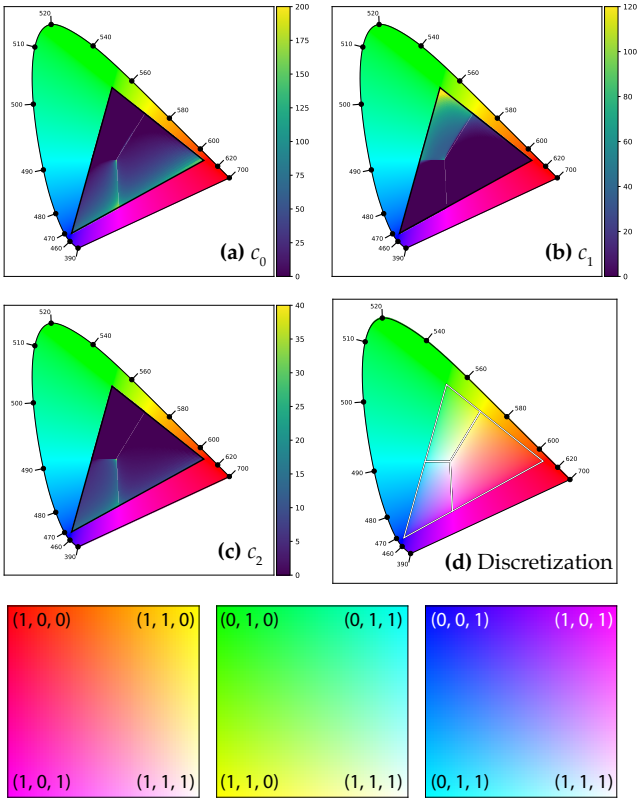
3.1. Optimization

Having decided on the model, we now turn to the task of computing the coefficients c_i for a given RGB value. We use the CERES solver [AMO] to do so, which internally relies on a variant of the Levenberg-Marquardt algorithm along with gradients of the objective (1) computed via forward mode automatic differentiation. The optimization rapidly converges in a few iterations from a starting guess of zero.

However, simply running the nonlinear solver for all input RGB values would not yield satisfactory results: because the solution is not always unique for saturated bright or very dark colors, drastic changes to the output spectra can result from small perturbations to the input color. Such transitions would cause visible discontinuities in a spectral rendering whenever a material is lit by an illumination spectrum other than the white point illuminant. The solution to this problem is simple: starting from a color value $\mathbf{c} = (r,g,b)$ of moderate brightness where the solution is stable (e.g. $\max\{r,g,b\} = 0.1$), we iteratively solve for brighter and darker colors $(\alpha r, \alpha g, \alpha b)$ in both directions, while using the solution of the previous iterate as a starting guess.

Another problem is that the nonlinear solver is far too costly to be invoked during texture lookups, hence the optimization must be carried out beforehand. Meng et al. [MSHD15] store precomputed spectra on a discretization of the spectral locus into square and triangular elements. As discussed in Section 2, this leads to a relatively complex implementation that we wish to sidestep by specializing to specific RGB color spaces.

This prompts the question of how the RGB domain should be discretized: Figure 3a-c show optimized coefficients c_0, c_1, and c_2 for maximally bright colors on the sRGB gamut, which reveals an interesting pattern: several ridges pass along straight lines connecting the white point to red, green, blue, and their complementary



(e) Discretization (unwarped)

Figure 3: *Coefficient maps and chosen discretization for sRGB*

colors cyan, magenta, yellow. These are locations where the spectra considerably change in shape–for instance, by switching from one central peak to two separate peaks. Motivated by this observation, we define three quadrilateral regions (Figure 3d) that are bounded by exactly these points and map a regular grid onto each one. The coefficients are very smooth within each region, and we found a discretization of 64×64 to be more than sufficient.

Implemented in this way, our method would produce a set of three coefficient maps requiring 144KiB of storage that encode the parameters of maximally bright (i.e. $\alpha := \max\{r,g,b\} = 1$) RGB colors. To handle darker colors, we could adopt the approach used by most previous work, which exploits the linearity of the mapping by fetching coefficients for normalized colors $(r,g,b)^T/\alpha$ and scaling the resulting spectra by α.

While the small footprint of a 2D tabulation is appealing, the focus on maximally bright colors is problematic: this approach maps dark colors to spectra that tend to be much more strongly peaked than is actually necessary (Figure 4). Secondly, it is often impossible to achieve zero error for a maximally bright RGB color value when working with large-gamut color spaces, which complicates the use of such a scheme.

These issues motivate our decision to extend the tabulation with an extra dimension that captures different values of $\alpha \in [0,1]$, where darker colors generally map to smoother spectra that achieve lower (or zero) error. Our final discretization then consists of three

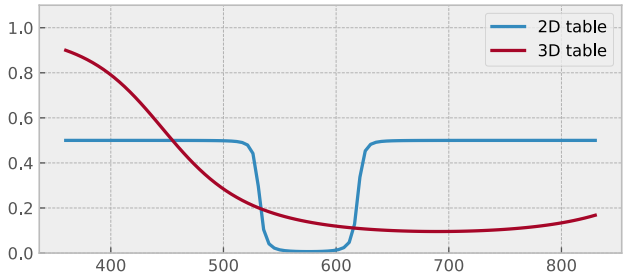


Figure 4: *Spectral plots corresponding to the color* `[.2,.2,.5]` *in ACES 2065-1, using a 2D **(blue)** and a 3D tabulation **(red)**. The latter maps darker colors to significantly smoother spectra.*

separate 3D cubes of resolution 64^3 requiring a total of 9 MiB of storage (Figure 6). Interestingly, this exceeds the size of all tables in prior work by a significant margin, yet this overhead is unproblematic: since the tables must only be accessed once at load time to transform RGB texels into model parameters (c_0, c_1, c_2), they do not cause additional memory traffic while rendering.

The last detail that must be discussed is how to discretize the newly added dimension: instead of a linear spacing, we use a mapping that dedicates a larger fraction of the resolution to very bright and very dark spectra where the parameters of our model vary more quickly (Figure 5). The i-th slice corresponds to Figure 3e, scaled by $\alpha_i := \texttt{smoothstep(smoothstep(i/63))}$. Algorithm 2 shows how to fetch spectrum coefficients from the resulting tabulation. Depending on the target color space, the complete optimization takes 10-60s on an Intel i7-6700K laptop CPU.

Algorithm 2 Lookup into precomputed tabulation

```
1:  function FETCH(color)
2:      # Find the largest component
3:      i = color.argmax()
4:
5:      # Normalize other two components
6:      color_norm = [
7:          color[(i + 1) % 3] / color.max(),
8:          color[(i + 2) % 3] / color.max(),
9:          color.max()]
10:
11:     # Trilinearly interpolated lookup
12:     return table[i].eval(color_norm)
```

4. Evaluation

In the following we compare our work to the upsampling methods by MacAdam [Mac35a], Smits [Smi99], Meng et al. [MSHD15], and Otsu et al. [OYH18].

Execution speed. We evaluated the speed of all methods on a simple test that involves upsampling a 4096× 4096 pixel floating point RGB texture to spectra with 16 wavelengths per pixel. Results can be found in Table 1. Our method is broken down into two steps: a preprocess where RGB tristimulus values are replaced by the coefficients for our parametric spectra, and the actual conversion.

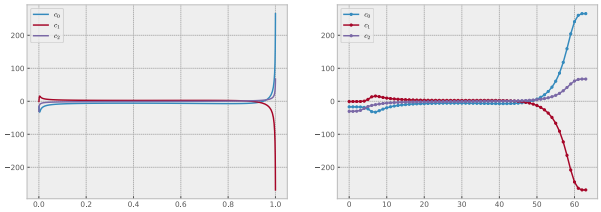


Figure 5: *The coefficients of our model change rapidly near $\alpha = 0$ and $\alpha = 1$ **(left)**, hence we use a discretization that increases the resolution in these regions **(right)**.*

Our method is significantly faster than the other methods. Only MacAdam's reconstruction could potentially be faster if performed during a preprocess. On the Knights Landing platform, we achieve good speedups with SSE and AVX, but for the AVX512 case the memory access and control logic around the reconstruction code begins to dominate, which diminishes the performance benefits of very wide vectorization.

Method	Core i7	Xeon Phi
MacAdam	1.508 03	9.160 01
Smits	2.326 72	11.9571
Meng	8.531 76	60.73
Otsu	2.609 84	14.216
Our (preprocess)	1.303 09	8.169 92
Our (scalar)	0.837 51	3.438 02
Our (SSE4.2)	0.209 26	0.882 43
Our (AVX)	0.107 90	0.437 30
Our (AVX512)	-	0.349 88

Table 1: *Evaluation speed (measured in seconds) when upsampling a 4K texture into a spectral representation with 16 wavelengths per pixel, on an Intel Core i7-5500U CPU @ 2.40GHz and an Intel Xeon Phi CPU 7210 @ 1.30GHz Knights Landing processor (both single threaded).*

Colorimetric properties. Figure 8 shows errors produced by the different upsampling methods, plotted in the CIE xy graph. Every pixel in the image shows the CIE76 ΔE difference between the original color $\mathbf{b}$ and the one corresponding to the upsampled spectrum, i.e. $\mathrm{rgb}(\mathrm{spec}(\mathbf{b}))$. The top row shows XYZ input normalized such that the maximum component is one. These are often infeasibly bright colors that need to be brought into the reflectance gamut, hence all methods show relatively large error. The second row is normalized such that the input color is scaled to half the brightness of the theoretical limit for reflectances [Mac35b] All input colors should be well represented by reflectance spectra here. Meng et al.'s method still shows some error near the boundaries of the spectral locus. The spectra are extremely peaked in these regions, which makes errors susceptible to details of the quadrature scheme, as well as the ability of approximating Dirac delta functions. The third row finally shows input in sRGB color space with maximum component equal to one, representing a common use case in practical usage. Our method is the only one with zero error in this case.

Our optimization process considers 3D input, i.e. it will create flatter spectra for colors with lower saturation, allowing it to out-

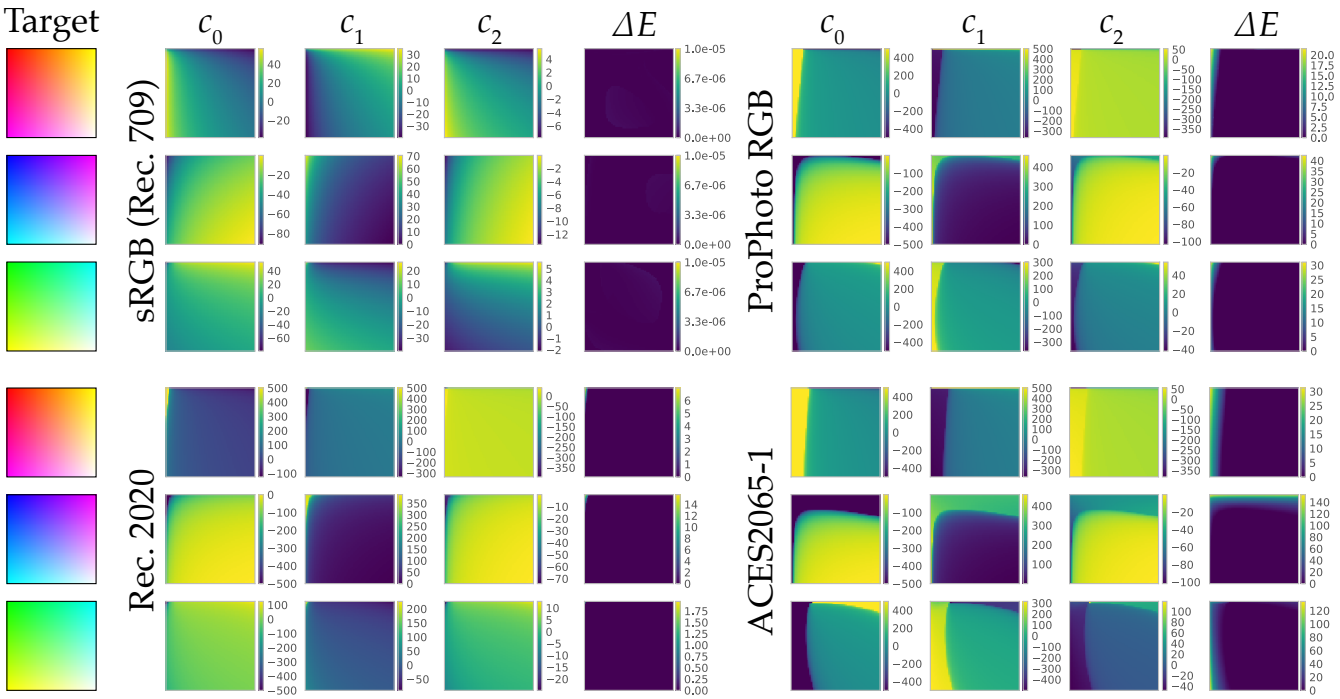


Figure 6: *Visualization of precomputed model coefficients for several widely used RGB color spaces (the $\alpha = 1/2$ slice is shown.)* ***(top left)****: The sRGB gamut is fully contained in the spectral locus, which leads to very smooth coefficients with color reproduction errors in the order of the machine precision.* ***(bottom left)****: The Rec. 2020 gamut is interesting because it touches the spectral locus, which leads to a localized error increase that is visible in the top left of all slices.* ***(right column)****: Both ProPhotoRGB and ACES2065-1 are large-gamut color spaces containing imaginary colors that lie outside of the spectral locus. Our model also supports such colors but gamut-maps them onto a realizable spectrum that minimizes the CIE ΔE error. The resulting change in behavior and increased errors are apparent in the plots.*

perform Meng et al.'s and Otsu et al.'s methods that are based on 2D precomputations (on a grid or a kd-tree, respectively).

MacAdam's method suffers from quantization issues due to the sharp edges of the box spectra coupled with the fixed resolution of our integration scheme. We include it for completeness, since MacAdam's spectra represent the limiting case of maximum color saturation. We note that they are usually not a good choice for rendering.

Figure 7 shows one potential issue with Meng at al.'s method: Since the spectra are only precomputed at a sparse set of points, they need to be interpolated to span the whole *XYZ* gamut. While this results in the correct color, the shape of the spectra can be oscillatory when interpolating between four spiky spectra. The comparison to Otsu et al.'s and our method shows that much smoother spectra representing the same color are possible in this case.

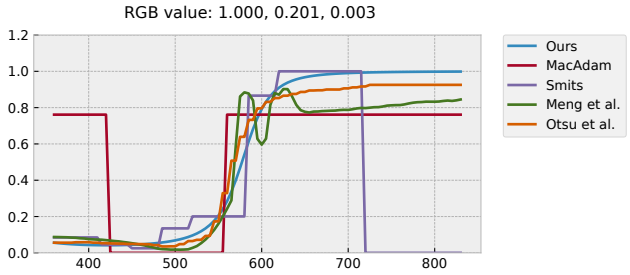
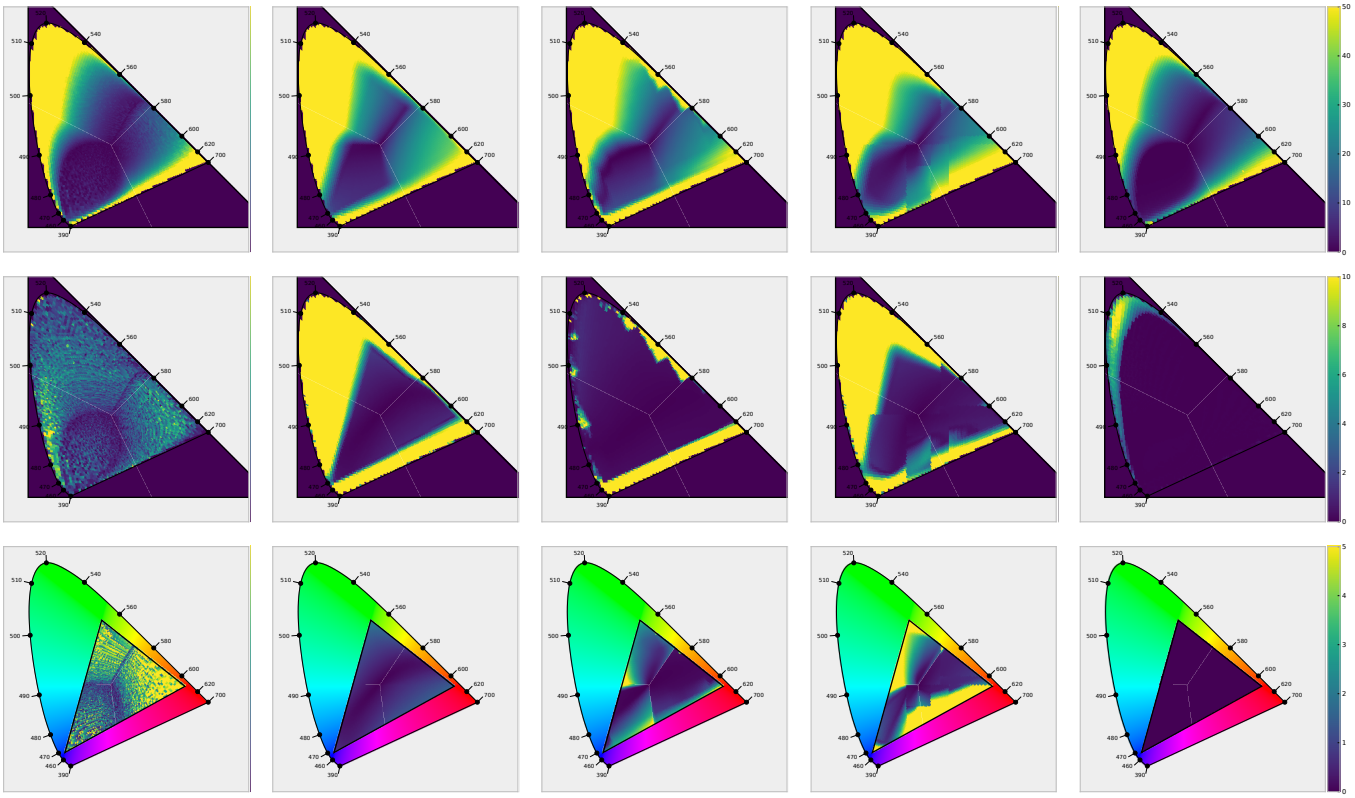


Figure 7: *Example spectrum where Meng et al.'s method shows clear interpolation artifacts: the four spectra which are used for interpolation here have distinctly different peaks which all show up in the non-smooth final result.*

Radiometric properties. Figure 10 shows three different RGB color ramp textures (in sRGB adapted to illuminant E) lit using CIE illuminants E and F2, generated using all previously discussed upsampling techniques.

We show our method in two variants: *Ours* upsamples the interpolated RGB values from the input texture ramp, which leads to output that is identical to the illuminant E-lit reference (first row of each plot in the left column). *Ours (int.)* transforms the RGB colors of the endpoints of the color ramp into coefficients (c_0, c_1, c_2) of our model and upsamples the interpolated coefficient representation instead. We emphasize that this is not how our method would typically be used, but include the result to show the relationship between the resulting color and its coefficient representation. We conclude that—within limits—it can be reasonable to interpolate coefficients instead of RGB values or spectra. Because of the nonlinear sigmoid transformation $S(\cdot)$, this is different from interpolation in RGB or in the spectral domain, but the smooth mapping



MacAdam [Mac35a] Smits [Smi99] Meng et al. [MSHD15] Otsu et al. [OYH18] Ours

Figure 8: *Round-trip errors between the input color and the one corresponding to the reconstructed spectra (CIE76 ΔE, an error of 2.3 would be a "just noticable difference' (JND)). The brightness of the input color in the top row is normalised such that the maximum XYZ component is unity. These spectra are gamut mapped to be a valid reflectance (i.e. bounded values in the interval $[0,1]$) which is visible in large round trip errors for all methods. The center row is scaled to half of MacAdam's theoretical limit on brightness. The bottom row shows the most common use case of upsampling sRGB colors. These are normalized to maximum component of one, which is different to the first row. Due to the 3D optimization approach, our method performs best in all cases and achieves zero error on the sRGB gamut.*

does not produce unexpected artifacts (jitter, non-monotonicity, or additional intermediate color transitions).

MacAdam's method suffers from substantial numerical jitter due to the sharp edges in the spectra. Otsu et al.'s method shows discontinuities when crossing kd-tree cell boundaries in the blue/magenta gradient, and when it is lit by illuminant F2, which models a fluorescent light source.

We also evaluated the shape of the resulting spectra and their behavior with indirect lighting. We selected several patches from the X-Rite ColorCheckerSG and simulated nine bounces of light on them by multiplying the spectra. The results are converted to sRGB and shown as color patches in Figure 9. MacAdam's spectra have a tendency to decay faster with multiple bounces due to their comparably low maximum value. Otsu et al.'s method achieves an excellent match to the spectrum in row 3, potentially because a similar spectrum was used as part of the method's training. However, the spectra produced for the other patches are very different in shape. In row 4, the method of Smits saturates at 1.0 and thus does not decay at all with multiple bounces. Our method yields results that are comparable to the other methods, but without artifacts such as overly fast decay (MacAdam) or sustaining saturation (Smits).

5. Conclusion

We presented a fast and practical technique to upsample tristimulus textures to full spectra suitable for spectral rendering. The proposed method works on input color with arbitrary gamut and represents spectra with a simple parametric model. This model produces smooth spectra where possible (low brightness) and transparently blends over to boxy spectra for high color saturation and brightness. This is possible because we optimize spectra for 3D input, not only on the 2D space of xy chromaticities such as previous work. We use three input parameters, i.e. an RGB texture converted to our coefficients maintains the exact same memory footprint. Run time evaluation for any given wavelength requires as little as six floating point operations and is trivially vectorized for modern instruction sets, and we demonstrate run time speedups over previous work of $15\times$ up to $150\times$.

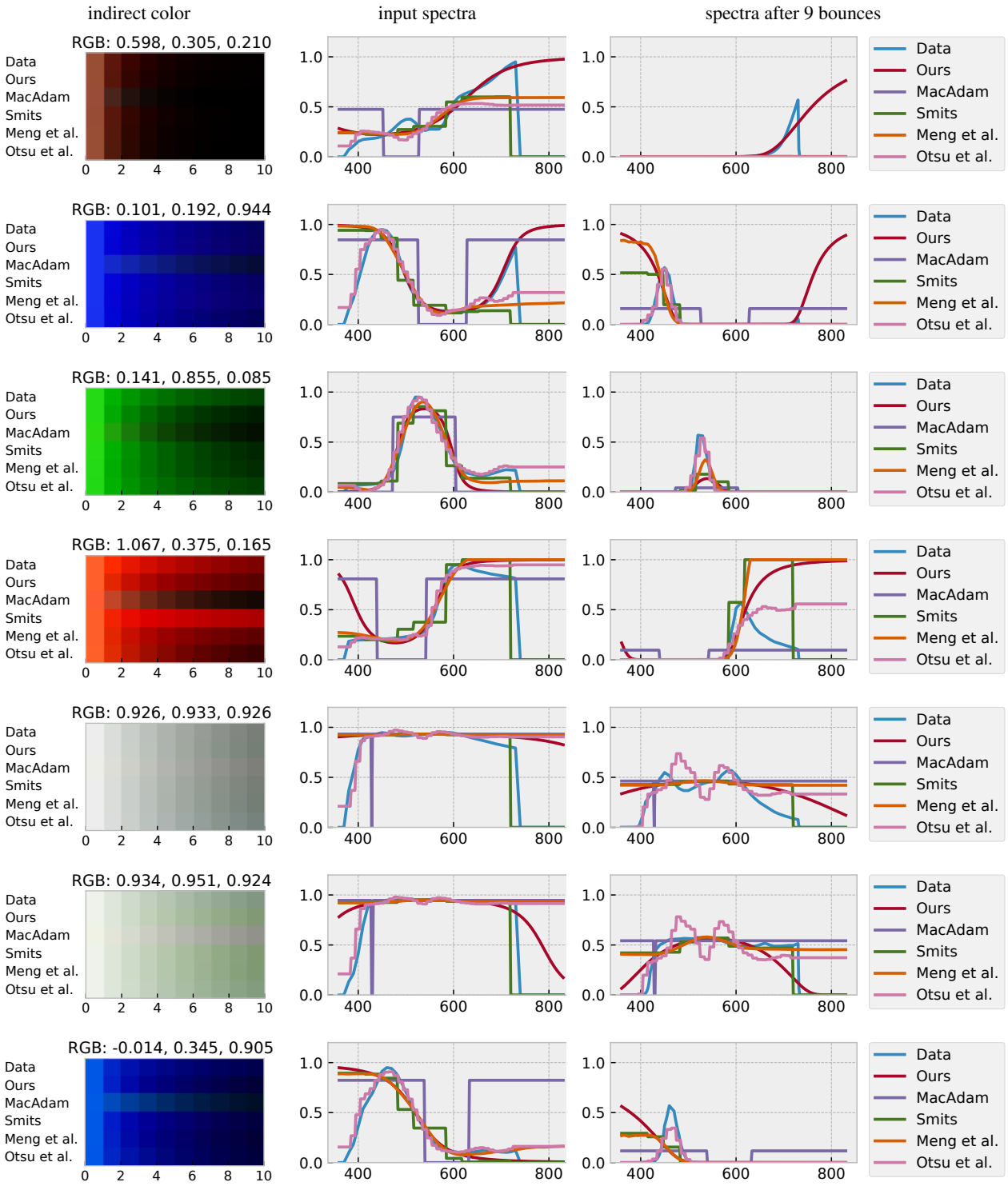


Figure 9: *Evaluation of indirect light: the input spectrum (lit by illuminant E) is multiplied by itself several times, simulating indirect lighting. The first row shows this directly on the input spectrum (obtained from the X-Rite ColorCheckerSG), the others first convert the spectrum to RGB and then back to a spectrum with the signified method, and the indirect lighting is simulated on this spectrum. The reflectance spectra are plotted in the middle and the output spectra after nine bounces are plotted on the right.*

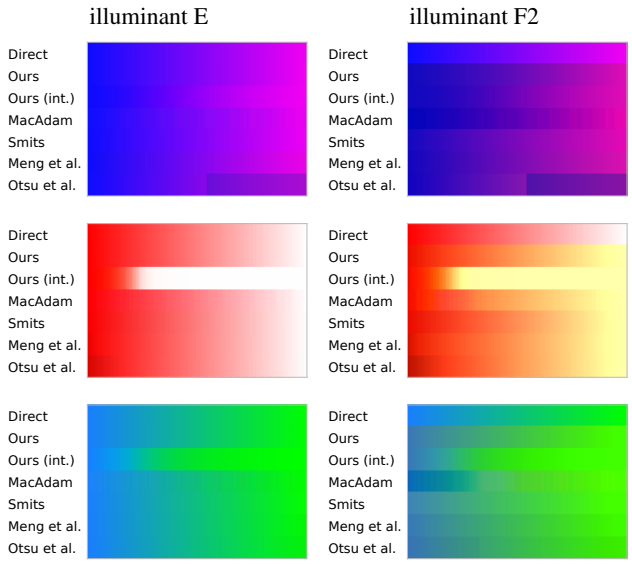


Figure 10: *RGB texture with a color ramp (directly visualized on the top row in each figure), rendered under illuminants E (**left**) and F2 (**right**). Ideally all methods should look the same (and equal to the top row for illuminant E). However, overly saturated and bright colors lead to clamped reflectance spectra in the range* $[0,1]$ *and thus to different RGB depending on the shape of the spectrum.*

References

[AMO] AGARWAL S., MIERLE K., OTHERS: Ceres solver. http://ceres-solver.org. 4

[BB17] BELCOUR L., BARLA P.: A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence. *ACM Transactions on Graphics 36*, 4 (July 2017), 65. 1

[DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graph. 35*, 1 (Dec. 2015), 9:1–9:13. 1

[FHF*17] FASCIONE L., HANIKA J., FAJARDO M., CHRISTENSEN P., BURLEY B., GREEN B.: Path tracing in production – part 1: Writing production renderers. In *SIGGRAPH 2017 Courses* (2017). 1

[IA99] ICART I., ARQUÈS D.: An illumination model for a system of isotropic substrate-isotropic thin film with identical rough boundaries. In *Rendering Techniques' 99*. Springer, 1999, pp. 261–272. 1

[Mac35a] MACADAM D. L.: Maximum visual efficiency of colored materials. *Journal of the Optical Society of America) 25*, 11 (1935), 361–367. 2, 5, 7

[Mac35b] MACADAM D. L.: The theory of the maximum visual efficiency of colored materials. *Journal of the Optical Society of America) 25*, 8 (1935), 249–249. 2, 3, 5

[MSHD15] MENG J., SIMON F., HANIKA J., DACHSBACHER C.: Physically meaningful rendering using tristimulus colours. *Proc. Eurographics Symposium on Rendering 34*, 4 (June 2015), 31–40. 3, 4, 5, 7

[OYH18] OTSU H., YAMAMOTO M., HACHISUKA T.: Reproducing spectral reflectances from tristimulus colours. *Computer Graphics Forum 37*, 6 (2018), 370–381. 3, 5, 7

[Pee93] PEERCY M. S.: Linear color representations for full speed spectral rendering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, pp. 191–198. 1

[Sch19] SCHRÖDINGER E.: Theorie der Pigmente größter Leuchtkraft. *Annalen der Physik 367*, 15 (1919), 603–622. 2

[Smi99] SMITS B.: An RGB-to-spectrum conversion for reflectances. *Journal of Graphics Tools 4*, 4 (1999), 11–22. 2, 5, 7

[WEV02] WARD G., EYDELBERG-VILESHIN E.: Picture Perfect RGB Rendering Using Spectral Prefiltering and Sharp Color Primaries. In *Eurographics Workshop on Rendering* (2002), The Eurographics Association, pp. 117–124. 1

[WVJH17] WERNER S., VELINOV Z., JAKOB W., HULLIN M.: Scratch iridescence: Wave-optical rendering of diffractive surface structure. *Transactions on Graphics (Proceedings of SIGGRAPH Asia) 36*, 6 (Oct. 2017). 1

[YHW*18] YAN L.-Q., HAŠAN M., WALTER B., MARSCHNER S., RAMAMOORTHI R.: Rendering specular microgeometry with wave optics. *ACM Trans. Graph. 37*, 4 (July 2018), 75:1–75:10. 1