Adapting Precomputed Radiance Transfer to Real-time Spectral Rendering

Karsten Schwenk[1], Tobias Franke[1], Timm Drevensek[1], Arjan Kuijper[1,2], Ulrich Bockholt[1,2] and Dieter W. Fellner[1,2]

[1]Fraunhofer IGD, Germany
[2]TU Darmstadt, Germany

Abstract

Spectral rendering takes the full visible spectrum into account when calculating light-surface interaction and can overcome the well-known deficiencies of rendering with tristimulus color models. We present a variant of the precomputed radiance transfer algorithm that is tailored towards real-time spectral rendering on modern graphics hardware. Our method renders diffuse, self-shadowing objects with spatially varying spectral reflectance properties under distant, dynamic, full-spectral illumination. To achieve real-time frame rates and practical memory requirements we split the light transfer function into an achromatic part that varies per vertex and a wavelength-dependent part that represents a spectral albedo texture map. As an additional optimization, we project reflectance and illuminant spectra into an orthonormal basis. One area of application for our research is virtual design applications that require relighting objects with high color fidelity at interactive frame rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Spectral rendering, i.e. lighting calculations that take the full visible spectrum into account, can be used to overcome the well-known deficiencies of rendering with tristimulus color models when it comes to accurate color reproduction [RP98]. Most work on the topic has been carried out with offline renderers in mind, but the larger accuracy of spectral rendering can also improve the results of real-time rendering approaches based on hardware-accelerated rasterization.

In this paper, we adapt the well-known precomputed radiance transfer (PRT) algorithm [SKS02] to real-time spectral rendering. Existing real-time spectral rendering methods assume a simple lighting environment, usually consisting only of a few directional-, point-, or spotlights. Established PRT methods, on the other hand, are able to handle complex lighting environments in real-time, but carry out their calculations in a RGB color space, which negatively affects color reproduction. Our approach combines both worlds and renders *diffuse*, *self-shadowing* objects with *spatially varying spectral reflectance* properties under *distant*, *dynamic*, *full-spectral* illumination.

The key idea of our method is to project the data into *two* orthonormal bases, one for the spectral domain and one for the directional domain. With this compression scheme, the spectral reflection calculation can be approximated by two dot products of coefficient vectors. This computation can be carried out efficiently in shader programs on programmable graphics hardware. As an optimization, we split the light transfer function into an achromatic part that varies per vertex and a wavelength-dependent part that represents a spectral albedo texture map. This ensures real-time frame rates and moderate memory requirements, but limits us to diffuse objects with self-shadowing (i.e. no interreflections).

The primary motivation for our work is to improve color correctness in the rendering pipeline of interactive virtual design applications. Such systems often run on desktop graphics hardware or even on mobile devices. Rendering techniques for these applications require high color fidelity as well as interactive frame rates. We show that our method can significantly improve color reproduction in rendered images when compared to RGB rendering. A performance penalty arises from storing and processing spectral data, but it is tolerable for most applications.

2. Related Work

Our work combines PRT and spectral rendering. We briefly review related work in these fields and denote its relevance to our paper.

Sloan et al. [SKS02] coined the term *precomputed radiance transfer*. The key innovation was to represent the radiance transfer function on geometry meshes as spherical harmonics coefficients and combine it at runtime with environment lighting in the same representation. Our work is directly based on this paper. For a review of more recent PRT methods we refer to [Ram09].

Point sampling spectra is seldom practical for spectral rendering because of the high sampling costs incurred. We use the approach taken by Peercy [Pee93] to compactly represent spectra. This method uses characteristic vector analysis to select a set of fixed orthonormal basis functions that can efficiently represent all spectral power distributions in a scene.

Spectral rendering is primarily used in offline rendering systems, and little research has been done on how to implement spectral reflection calculations in the context of real-time rendering. One early contribution is Johnson and Fairchild's work [JF99]. They extended the OpenGL pipeline to perform reflection calculations per wavelength and to interactively simulate fluorescence. They focus on faithful color reproduction like we do, but they are limited to OpenGL's simple point/spot/directional light sources.

Ward et al. introduced *spectral prefiltering* to improve the color reproduction in RGB-based rendering pipelines [WEV02]. With this method all material spectra are premultiplied by a dominant light source spectrum of the scene and projected into a tristimulus color space. Then the dominant sources are replaced by white sources of equal intensity and sources with different spectra are modified to account for the prefiltering. The resulting light and material colors can be used in any tristimulus renderer without further overhead, making the method well suited for real-time rendering. Unfortunately, the method requires a scene-specific preprocessing step and needs a dominant light source spectrum in the scene. If light sources that deviate from this dominant spectrum are present, the accuracy of the method declines, because reflection calculations are not really carried out with spectra. In contrast, our method allows for free recombination of objects and illumination settings without the need for a combination-specific preprocessing. As a true spectral rendering method it has also higher rendering costs.

More recently, Duvenhage developed a pipeline for spectral rendering on programmable graphics hardware [Duv06]. The pipeline is based on a manually factored representation of the BTF into a component that varies only with surface parameterization and a component that models a low-resolution spectral BRDF. Compared to their system, we cannot handle non-diffuse BRDFs. On the other hand, we support spectral albedo texture maps, complex self-shadowing geometry, and spectral environment map lighting.

Lindsay et al. render intererence effects by precalculating the view-dependent spectral surface response of surfaces to hemisphere lighting in terms of a spherical harmonic basis [LA05]. Their approach is similar to ours, but we focus on faithful color reproduction with spectral albedo maps. Hence we employ a different spectral representation. Our response is diffuse, which allows us to let it vary with high frequency over the surface (per texel) and specify only a low-frequency achromatic transfer vector (per vertex). Their response is view-dependent but needs a full spectral transfer vector.

3. Our Approach

3.1. Foundations

The foundation of our method is the local reflectance integral, explicitly formulated with spectral quantities:

$$L_o(\omega_o,\lambda) = \int_{\mathcal{H}(n)} f_r(\omega_i,\omega_o,\lambda) L_i(\omega_i,\lambda)\cos\theta_i \mathrm{d}\omega_i. \quad (1)$$

This equation is defined for each surface point in a scene and relates the outgoing spectral radiance L_o in direction ω_o to the incident spectral radiance L_i from direction ω_i. $\mathcal{H}(n)$ is the Hemisphere defined by the surface normal n. We make two simplifications. Firstly, we consider only perfectly diffuse surfaces, so $f_r(\omega_i,\omega_o,\lambda) = f_r(\lambda)$. Secondly, we only consider self-shadowing (no indirect light transport) and distant environment map lighting. So $L_i(\omega_i,\lambda) = L_e(\omega_i,\lambda)V(\omega_i)$, where L_e is the spectral radiance stored in the environment map and V is the visibility function defined for each surface point and direction. This yields:

$$L_o(\lambda) = f_r(\lambda) \int_{\mathcal{H}(n)} L_e(\omega_i,\lambda)V(\omega_i)\cos\theta_i \mathrm{d}\omega_i. \quad (2)$$

Note that in general this approach results in a more accurate color reproduction because the local reflectance is evaluated with spectral quantities instead of colors. Also, in general, this is more accurate than spectral prefiltering if the individual light samples deviate from the dominant spectrum used in the prefiltering step.

After the reflected spectral radiance arriving at a pixel is calculated, it can safely be converted into a CIE XYZ [Int09] color for further processing:

$$M_o = \int_{400nm}^{700nm} L_o(\lambda)\overline{m}(\lambda)\mathrm{d}\lambda. \quad (3)$$

This equation is applied to each color component $M_o \in \{X,Y,Z\}$ using the corresponding color matching function $\overline{m} \in \{\overline{x},\overline{y},\overline{z}\}$.

After the CIE XYZ color has been computed, one has to be careful in order not to jeopardize the increased accuracy of the spectral reflection calculation during post processing.

In our system, we simulate chromatic adaption (white balance) using the Bradford transform to relate the white point of the rendered scene to the viewing conditions of the display [FS00]. For the images in this paper we used the D65 white point of sRGB. Then we convert the XYZ colors to linearized sRGB space, apply a sigmoid tone mapping operator, and correct for the display's gamma curve.

3.2. Basis for spectral domain

Point sampling the spectral domain to approximate the integral in Equation (3) can be very expensive in the presence of spiky spectra that demand a high sampling rate. We have evaluated several compression methods for spectral data and found the approach taken by Peercy [Pee93] to be best suited for our algorithm. Peercy's Linear Model tries to find an optimal (in terms of RMS-error) finite-dimensional orthonormal basis for a given set of spectra. To compute this basis, we sparsely sample the contents of all spectral reflectance maps in a scene and all potential environment maps and assemble these spectra into the columns of a matrix. The basis vectors are found by performing a SVD of this matrix. As was observed by Peercy, few basis vectors are usually needed to accurately represent the spectra. Even under difficult lighting conditions, e.g. with the CIE F-series that have very spiky spectra, we didn't need more than 8 coefficients to get a result indistinguishable from 5nm point sampling with our test cases.

3.3. Basis for directional domain

In the directional domain we restrict ourselves to low-frequency lighting environments. Therefore the spherical harmonics basis [SKS02] can be used. However, we would like to point out that the derivations in the following section only require the basis to be orthonormal, so in other scenarios another basis can be used (e.g. an orthonormal Wavelet-basis on the sphere).

3.4. Spectral radiance transfer

Our goal is to approximate Equation (3). By expanding L_o according to Equation (2), setting $R(\lambda) = f_r(\lambda)\overline{m}(\lambda)$ (i.e. premultiplying the albedo by the color matching functions), and introducing the transfer function $T(\omega_i) = V(\omega_i)\cos\theta_i$ we get:

$$M_o = \int_{400nm}^{700nm} R(\lambda) \int_{\mathcal{H}(n)} L(\omega_i,\lambda) T(\omega_i) \mathrm{d}\omega_i \mathrm{d}\lambda. \quad (4)$$

R and L are projected into the spectral basis $\phi_i(\lambda)$:

$$R(\lambda) \approx \sum_i^m \hat{R}_i \phi_i(\lambda), \quad L(\omega_i,\lambda) \approx \sum_j^m \hat{L}_j(\omega_i)\phi_j(\lambda).$$

T and the $\hat{L}_j$ are projected into the directional basis $\psi_i(\omega)$:

$$T(\omega) \approx \sum_k^n \tilde{T}_k \psi_k(\omega), \quad L(\omega,\lambda) \approx \sum_j^m \sum_l^n \tilde{L}_{jl}\phi_j(\lambda)\psi_l(\omega).$$

By replacing R, T, and L in Equation (4) with their approximations and using the fact that the basis functions are orthonormal in their domains we obtain:

$$M_o \approx \sum_j^m \sum_k^n \hat{R}_j \tilde{L}_{jk} \tilde{T}_k = \sum_j^m \hat{R}_j \left(\sum_k^n \tilde{L}_{jk} \tilde{T}_k \right). \quad (5)$$

This equation can be evaluated efficiently on programmable graphics hardware. The achromatic transfer vector $\tilde{T}$ is stored per vertex, just like in classical PRT. $\tilde{R}$ is stored in an texture array with m layers. Note that each entry for $\hat{R}$ has three components, one for each color matching function that it was premultiplied by; they are stored in the RGB channels. Finally, $\tilde{L}$ is passed to the shader as a constant array of size $n \times m$. The inner sum in Equation (5) is computed in the vertex shader. This yields the spectral irradiance $\hat{E}$ per vertex in our spectral basis. In the fragment shader, we compute the inner product $(\hat{E}.\hat{R})$ for each color matching function, which gives the reflected spectral radiance projected into CIE XYZ. The orthonormality of the spectral basis allows us to combine the calculation of the reflected spectral radiance and the projection into the color space into a single inner product.

4. Results

For the renderings in this paper, we generated spectral environment maps for the CIE standard illuminants. We used 8 characteristic vectors as basis for the spectral domain and the first five bands of spherical harmonics for the directional domain. All images were taken using the 1931 CIE 2° observer.

Figure 1 shows a replication of the METACOW [FJ04] image rendered with our spectral PRT method. The left half of each cow has the spectral reflectance of the GretagMacbeth Color Checker chart [Mun09], while the right half is a metameric black that has been calculated to maximize color difference under Illuminant A. For the rendering we simply mapped a spectral albedo map onto a plane. Note how the spectra are metamers under illuminant D65, but resolve to different colors when viewed under illuminant A and illuminant F2. Naive RGB rendering is not able to capture this effect and would render all objects in the same color.

Figure 2 shows a comparison of our technique with a classical RGB-PRT rendering. The spectral albedo map contains the 24 spectra of the GretagMacbeth ColorChecker chart. The illuminant is F4. The error is plotted using the CIE94 color difference model [MS95]. Even with these relatively smooth reflection spectra color reproduction is poor with RGB rendering for saturated colors under a spiky light source spectrum.

Compared to RGB-PRT our method has an overhead on CPU and GPU proportional to the number of basis functions used for the spectral domain. In practice, our method was 2-4 times slower than regular RGB-PRT. The exact factor

Figure 1: *A replication of the Metacow image rendered with our spectral PRT method. Left: Illuminant D65, center: Illuminant A, right: Illuminant F2. RGB rendering would not resolve the metamers under illuminants A and F2.*

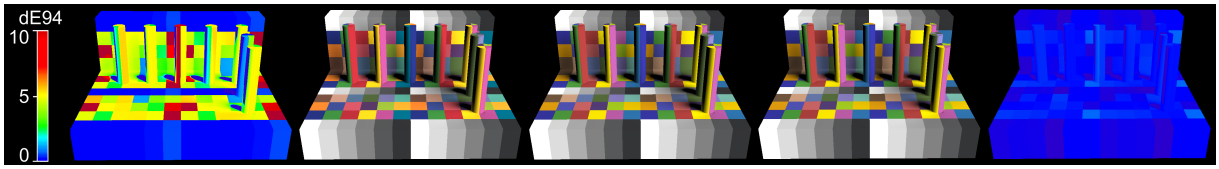

Figure 2: *A model mapped with the patches of the Macbeth ColorChecker under illuminant F4. Center: Reference solution (5nm point sampling). To the right: Our spectral rendering method and CIE94 ΔE_{94} difference to reference. To the left: RGB rendering and difference to reference. A ΔE_{94} under 2 contains an almost unseeable color variance, a ΔE_{94} of 5 is clearly noticeable, but the two colors are still similar, a ΔE_{94} above 5 is seldom tolerated.*

depends on the characteristics of the scenario. The model in Figure 2 was rendered at 580 Hz with Spectral PRT and at 1305 Hz with RGB-PRT.

5. Conclusion and Future Work

We presented a PRT algorithm for real-time, full-spectral rendering on modern GPUs and showed that the method can significantly improve color reproduction in rendered images at reasonable costs.

The current implementation has two restrictions that we want to address with future work. Firstly, we would like to include multiple light bounces to maintain accuracy during interreflections. Secondly, we are working on integrating support for glossy BRDFs into our system.

References

[Duv06] DUVENHAGE B.: Real-time spectral scene lighting on a fragment pipeline. In *SAICSIT '06* (2006), South African Institute for Comp. Scientists and IT, pp. 80–89. 2

[FJ04] FAIRCHILD M. D., JOHNSON G. M.: Metacow: A public-domain, high-resolution, fully-digital, noise-free, metameric, extended-dynamic-range, spectra test target for imaging system analysis and simulation. In *Color Imaging Conference'04* (2004), pp. 239–245. 3

[FS00] FINLAYSON G. D., SÜSSTRUNK S.: Spectral sharpening and the bradford transform. In *Proc. Color Imaging Symposium (CIS 2000)* (2000), pp. 236–243. 3

[Int09] INTERNANIONAL COMMISION ON ILLIMINATION: CIE 1931 standard colorimetric observer data. http://www.cie.co.at/, 2009. 2

[JF99] JOHNSON G. M., FAIRCHILD M. D.: Full-spectral color calculations in realistic image synthesis. *IEEE Comput. Graph. Appl. 19*, 4 (1999), 47–53. 2

[LA05] LINDSAY C., AGU E.: Spherical harmonic lighting of wavelength-dependent phenomena. In *Proc. of Eurographics* (2005), pp. 121–124. 2

[MS95] MCDONALD R., SMITH K. J.: CIE94 - a new colour-difference formula. *Journal of the Society of Dyers and Colourists 111*, 12 (1995), 376–379. 3

[Mun09] MUNSELL COLOR SCIENCE LABORATORY: Spectral reflectance of macbeth color checker patches. http://www.cis.rit.edu/mcsl/, 2009. 3

[Pee93] PEERCY M. S.: Linear color representations for full speed spectral rendering. In *SIGGRAPH '93: Proceedings* (1993), ACM, pp. 191–198. 2, 3

[Ram09] RAMAMOORTHI R.: Precomputation-based rendering. *Found. Trends. Comput. Graph. Vis. 3*, 4 (2009), 281–369. 2

[RP98] ROUGERON G., PÉROCHE B.: Color fidelity in computer graphics: A survey. *Computer Graphics Forum 17* (1998), 1067–7055. 1

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings* (2002), ACM, pp. 527–536. 1, 2, 3

[WEV02] WARD G., EYDELBERG-VILESHIN E.: Picture perfect RGB rendering using spectral prefiltering and sharp color primaries. In *EGRW '02* (2002), Eurographics Association, pp. 117–124. 2