A tool to create illuminant and reflectance spectra for light-driven graphics and visualization

Steven Bergner, Mark S. Drew, Torsten Möller

{sbergner,mark,torsten}@cs.sfu.ca, http://gruvi.cs.sfu.ca/
Graphics, Usability, and Visualization Lab
Simon Fraser University

Full spectra allow the generation of a physically correct rendering of a scene under different lighting conditions. In this paper we devise a tool to augment a palette of given lights and material reflectances with constructed spectra yielding specified colors or spectral properties such as metamerism or objective color constancy. We utilize this to emphasize or hide parts of a scene by matching or differentiating colors under different illuminations. These color criteria are expressed as a quadratic programming problem, which may be solved with positivity constraints. Further, we characterize full spectra of lights, surfaces, and transmissive materials in an efficient linear subspace model by forming eigenvectors of sets of spectra and transform them to an intermediate space in which spectral interactions reduce to simple component-wise multiplications during rendering. The proposed method enhances the user's freedom in designing photo-realistic scenes and helps in creating expressive visualizations. A key application of our technique is to use specific spectral lighting to scale the visual complexity of a scene by controlling visibility of texture details in surface graphics or material details in volume rendering.

Categories and Subject Descriptors: I.3.7 [**Three-Dimensional Graphics and Realism**]: Color

General Terms: Spectral light and reflectance design, Linear spectral color models, Quadratic programming, Optimization.

Additional Key Words and Phrases:

1. INTRODUCTION

Light interacting with matter and its processing by the human visual system is the basis of imaging. In graphics it is common to simply use RGB values for all interactions, although it is well known that such a coarse approximation of full spectra can lead to disastrous errors [Johnson and Fairchild 1999]. Hence, for both surface graphics, including transmissions, as well as volume graphics, we should be using a good approximation of spectra.

Spectra in the context of this paper will refer to both light as a spectral power distribution (SPD) over different energy levels (wavelengths) of the electromagnetic field, as well as wavelength dependent reflectance of the material. The latter arises from the fact that a material reflects or re-emits different wavelengths of incoming light to different degrees. Both light SPD and reflectance are modeled as scalar functions of wavelength and their basic interaction can be described as a product of the two, resulting in a reflected spectrum (color filtering). Additional effects involve shifts in emission of energy towards lower wavelengths (fluorescence) or at later points in time (phosphorescence). However, in the following we will consider color filtering alone, since this is the dominant effect if typical current RGB based photo-realistic scenes are converted and enhanced within a spectral rendering framework.

The field known as physically-based rendering aims at producing photo-realistic imagery [Pharr and Humphreys 2004]. Employing an appropriate algorithm that renders a convincingly realistic impression in practice is only a

part of the solution. First, we need a synthetic scene that contains sufficient detail and has a close correspondence to a physically plausible setting. Here one can distinguish three aspects: geometric modeling of shapes or density distribution of volumes, determining material appearance by setting reflection properties, and configuring the lighting of the scene. When switching the light model from RGB (red, green, blue) components to a spectral representation, the geometric modeling remains unaffected. Also, directional dependence of material shading properties as it is expressed by a bi-directional reflection distribution function (BRDF) can still be used. However, modeling the wavelength dependence of the BRDF as well as the light requires a new tool replacing the classical color picker.

While in the classical approach the reflectance or the light spectra are chosen separately, what we often need to model is the result of their interaction. I.e., we would like to input the resulting light-reflectance interaction as a constraint into our modeling system and have the system choose proper materials and lights.

The design tool devised in this paper is not limited to picking specific colors for certain light-material combinations. As well as that, it is aimed at taking advantage of certain effects that are inherent to the use of spectra. They are based on the notion of *metamers* — materials that look the same under one light, but may have clearly distinguishable color under another. In typical color matching problems metamerism is increased to make materials match under different lights. Our goal is to ensure metamerism only for one light, but to exploit the visual effect when the reflectances do not match if illuminated with a different light source. This can be employed to control visibility of additional detail in a scene as will be shown for a surface graphics example in § 4.3.

Another effect is that of a *metameric black* [Wyszecki and Stiles 1982] a surface spectral reflectance function that produces a zero RGB triple under a particular light. Under another light the surface appears colored, not black. Highly (but not completely) transparent materials tend to virtually disappear from a scene when they turn black. Another light that is reflected brightly may bring their appearance back to the user's attention. The question of how to incorporate such behavior into the design of spectra is the subject of § 3.

The obverse of the situation is one in which a user controls the appearance of a scene by changing the lighting. In a design involving metameric blacks as the difference between reflectances those materials retain their color as the light changes — we refer to this as *objective color constancy*. Then clearly, if one material is designed to change appearance, as the lights change, while other materials stay the same, we have a means to guide the user's attention, which can be used for the exploration of data sets.

Sampling the visible spectrum from 400 nm to 700 nm in 10 nm steps results in a 31-component spectral representation. Instead of our usual 3-vector component-wise multiplication, we have an order of magnitude more computational work. In a raytracing scenario where we may have billions of light-material interactions this will be very computationally expensive. Similar computational costs arise for a raycasting setting with many volume samples taken along each ray. Hence, we require a much more compact representation, which preserves the benefits of full spectra. Fortunately, a representation is available that accurately represents full spectra using about twice the number of components of tri-stimulus representations [Drew and Finlayson 2003] (5 to 7 basis coefficients here). Moreover, the new representation has the virtue that interactions can again be carried out using a component-wise multiplication, which we call a *spectral factor model*: instead of simple component-wise multiplication of RGB, we substitute a novel simple component-wise multiplication of basis coefficients.

The following section examines specifics of linear color models, concluding in an efficient representation for spectra. Our design method is proposed in § 3. There, different criteria are introduced and expressed as a least-squares problem. In § 4 an example scenario is described and the effects of the different criteria are explained. In addition, an application of the design framework to spectral volume rendering is discussed. The contributions of the approach are summarized in § 6 by providing a discussion of limitations and possible future directions. Supplementary material to the paper contains Matlab code to perform all design steps as well as a Java implementation. The material may be obtained at `http://www.cs.sfu.ca/gruvi/Projects/Spectral_Engine`.

2. RELATED WORK

A number of spectral rendering frameworks have been compared by Devlin et al. [Devlin et al. 2002], indicating a lack of open source solutions which has recently been changed by PBRT [Pharr and Humphreys 2004] only requiring a minor source modification to allow for rendering with a spectral color model. For application of the spectral rendering

concept in volume visualization see [Noordmans et al. 2000; Bergner et al. 2002; Abdul-Rahman and Chen 2005; Strengert et al. 2006] concerning the general use of spectra. In particular, [Abdul-Rahman and Chen 2005; Strengert et al. 2006] improved the accuracy of selective absorption by employing Kubelka-Munk theory that has previously been used [Gondek et al. 1994] to improve realism of renditions of oil paintings. For that spectral BRDFs for layered paint are acquired by analyzing simulated micro-structure as a pre-processing step. This is a promising extension to analytic models for interference colors or diffraction [Stam 1999; Sun et al. 2000]. A generative model to produce realistic impressions of human skin [Donner and Jensen 2006] also considers cross-effects between directional and energy dependence of reflectance. However, in the majority of appearance models the terms for directional and energy dependence of the reflectivity can be considered independently. Hence, we will in our design method concentrate on the wavelength dependence of the appearance only.

2.1 Previous approaches to constructing spectra

In this work we consider a method to obtain spectra to equip a synthetic scene according to certain appearance criteria. Spectral representations of light have their origin in more accurate models of physical reality. Hence, if the required materials or lights are available, a first choice would be to determine the reflectance or emission spectra by measuring them, e.g., via a spectrometer (see [Matusik et al. 2003]). Such measurements and imagery can be used to learn about chemical constituents of a light emitting or reflecting object of inspection. For instance, one can learn about the gas mixtures of stars via their emitting spectra, or one can find out about different vegetation on the surface of our planet using knowledge of distinct reflectances of plants and soils. When looking for sampled spectra online one typically finds graphs, but not the underlying data[1].

There is also a history of work on estimating spectral information from color filter array data, as used inside digital cameras or simply based on RGB images (e.g., [Drew and Funt 1992]). To resolve the problem of increased dimensionality of the spectral space over the measured components (usually three), assumptions are included about the illumination as well as the color filters. Both of these may be controlled as in [Farrell et al. 1999] who describe a system to obtain spectral approximations from digital photographic recordings of artwork.

For spectra already measured, one may also pick spectra from a database. [Wang et al. 2004] enhance this selection, by automatically choosing a neighborhood of 8 spectra surrounding a point of user specified chromaticity. These are linearly interpolated to produce an artificial spectrum close to physical spectra. Also, they use Bouguer's law to create additional spectra.

But a sufficiently large database may not be available to contain solutions satisfying all design constraints. Also, only using real physical spectra might not be a requirement in a computer graphics setting, which could instead also benefit from the creation of completely artificial spectra. Since linear basis functions are a successful model to represent measured spectra, they are the common choice to form a basis for modeling new spectra. Previous choices include delta functions at distinct wavelengths [Glassner 1989], boxes covering certain bands, and exponential functions, such as Fourier or Gaussian [Sun 1999]. These approaches produce mixtures of three functions for red, green and blue (RGB). To obtain flatter spectra [Smits 1999] also includes complementary colors (CMY).

All of the above methods to create artificial spectra have one common problem: they consider the design of a reflectance for a given color without considering the illumination spectrum. Such a design essentially only applies in controlling the appearance of self emitting objects. But for typical materials it is only the reflected spectra that we can see. These are related to the light spectrum, via a product with the wavelength dependent reflectance, but they are not the same. Thus, the color of a surface should indeed be chosen with respect to the reflected spectrum, but what really needs to be assigned in the scene description is a reflectance and a light.

This observation is the main motivation for our design method. The second main difference over previous methods is that we consider a design of an entire palette of several reflectances and lights instead of just single combination colors. This allows us to take effects of combined material appearance or lighting transitions into account. In the following,

[1] Using a vector paint program, such as Inkscape `http://www.inkscape.org`, one can trace the graphs. The node coordinates of the simplified and corrected path can be read from SVG and transformed and re-sampled to a fixed sampling distance using other numerical tools, such as Matlab or Octave. This may not be as accurate as the original data, but it is considerably more exact than eye-balling tri-stimulus color components.

we will provide some background on linear color models, leading to a choice of basis for efficient component-wise illumination computations, called the spectral factor model. The design method and its description in § 3, however, are independent of a particular choice of linear model.

2.2 Linear Color Models and a Spectral Factor Model

To obtain a low-dimensional representation of lights and surfaces, the most accurate representation that accounts best for variance is a principal component representation (see [Peercy 1993] for its use in surface graphics). However, for such an m-component representation, every light interaction will necessitate an $m \times m$ matrix multiply (this will be explicated below in Eq. 7). The method in [Drew and Finlayson 2003] obviates this via a so-called 'sharpening' transform. Motivated by human color vision, sharpened camera sensors are now the norm in digital color cameras (see, e.g., the sRGB space [Anderson et al. 1996]). Sharpening is a simple $m \times m$ preprocessing step for putting all calculations into a basis subspace that is a linear transform away from the original sensor space [Finlayson et al. 1994]. The idea of this transform is to make color sensors more like delta functions.

For our purposes, we wish to 'sharpen' the *basis set* for spectra. The optimal basis set to use is that derived from *color signals*, i.e., products of lights and reflectances [Drew and Funt 1992]. Then a preprocessing step of a constrained optimization can deliver the best basis space in which to work [Drew and Finlayson 2000]. If we choose to use fluorescent lighting, then we can develop a 'specialized' basis set that best describes the spike-like behavior of the spectra involved. In either case, the advantage is that, while we lose none of the expressive power of a principal component basis, when light participates in interactions the result in the subspace is well-modeled via a simple spectral factor model.

We describe the results of using a factor model in § 2.3. Rendering proceeds using coefficients with respect to this new basis set. One application that gains great benefit from such a compact model is volume rendering. Typically, the sampled values of a volume are assigned colors and opacities by means of a *transfer function*. Instead of directly assigning colors, we assign reflectance spectra. As rays are cast from the camera through the volume they accumulate the light that is reflected from each voxel, according to the opacities in the volume. Our approach of "post-illumination" [Bergner et al. 2002] modifies this projection to allow for a post-rendering re-illumination of the spectral image using different lights. Thus, a new form of real-time user interaction is created by allowing the scene to be manipulated via changing the light source (see § 4 for further examples). This principle also extends to surface graphics.

2.2.1 *Linear Color Models* .

Linear representations for spectra of lights and reflectances are attractive for rendering complex scenes for several reasons. Firstly, all illumination calculations can be performed in a linear subspace of reduced dimensionality; and the basis can be specialized for a set of representative spectra, thus improving accuracy. In general, each illumination computation in the linear subspace implies a matrix multiplication. The following discussion will construct this illumination matrix $\mathbf{R}$. In the special case of the *spectral factor model* the matrix is diagonalized, reducing the computation again to a simple componentwise multiplication, as detailed in § 2.3.

The basic idea of using a linear model is to describe a spectrum $C(\lambda)$ (e.g., a color signal [Wyszecki and Stiles 1982] formed from the product of light and surface spectral reflectance functions) by a linear combination of a set of basis functions B_i weighted by coefficients c_i:

$$C(\lambda) = \sum_{i=1}^{m} c_i B_i(\lambda). \tag{1}$$

The choice of basis functions can be guided by different criteria. [Marimont and Wandell 1992] discuss different approaches to finding a basis that minimizes *perceivable* errors in the sensor responses. [Peercy 1993] devised a framework for using linear color models in illumination calculations. The quality of a particular basis can be summarized as the tradeoff between accuracy and computational complexity. There are two different approaches to this issue. We can form *specialized bases* tailored to the particular set of spectra in a scene. Then these spectra have only minimal error when projected to the subspace spanned by the linear model. Spectra differing from the prototype spectra may have larger error from projection. Alternatively, *general bases* are suitable for a wider set of spectra. For instance, using exponential functions, such as a Fourier basis, only assumes some degree of smoothness of the modeled spectra.

Certainly, all spectra that are smooth enough will be well represented; however, a Fourier representation may have negative coefficients for valid physical (i.e., nonnegative) spectra, making the model problematic to use in a hardware implementation.

In order to computationally model interactions between spectral power distributions (SPDs) we represent the continuous function $C(\lambda)$ as a discrete vector. A *full spectrum* then consists of equidistant point samples taken over the visible range from 400 nm to 700 nm at 10 nm intervals forming a vector $\vec{C} \in \mathbb{R}^{31}$. The basis in Eq. 1 then becomes a $31 \times m$ matrix $\mathbf{B}$ comprised of the set of m basis vectors $\vec{B}_i$ and the coefficients c_i become a vector $\vec{c}$ approximating Eq. 1 as

$$\vec{C} \;=\; \sum_{i=1}^{m} c_i \vec{B}_i \;=\; \mathbf{B}\vec{c}. \tag{2}$$

Modeling illumination we will restrict ourselves to non-fluorescent interactions — no energy is shifted along different wavelengths of the spectrum. Hence, the observed reflected color spectrum equals a component-wise product of the two SPDs. We will use $\mathbf{diag}(\vec{S})$ as a diagonal matrix composed of the elements of $\vec{S}$ to define a component-wise multiplication operator $*$ between $\vec{E}$ and $\vec{S}$:

$$\vec{C} \;=\; \mathbf{diag}(\vec{E})\,\vec{S} \;=\; \mathbf{diag}(\vec{S})\,\vec{E} \;=\; \vec{E} * \vec{S} \;=\; \vec{S} * \vec{E}. \tag{3}$$

The coefficients $\vec{c}$ for a spectrum $\vec{C}$ can be obtained via the pseudo-inverse $\mathbf{B}^+$ of $\mathbf{B}$:

$$\vec{c} \;=\; (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\vec{C} \;=\; \mathbf{B}^+\,\vec{C}. \tag{4}$$

The spectra forming $\vec{C}$ can also be expressed in the linear subspace as $\vec{S} = \sum_{k=1}^{m} s_k \vec{B}_k$ and similarly $\vec{E}$. We combine the previous equations:

$$c_i \;=\; \vec{B}_i^{+}\,(\sum_{j=1}^{m} e_j \vec{B}_j * \sum_{k=1}^{m} s_k \vec{B}_k). \tag{5}$$

The two operands inside the parentheses are $\vec{E}$ and $\vec{S}$ from Eq. 3. The result of the spectral multiplication (in the *full* space $\mathbb{R}^{31}$) is then projected back onto the basis functions, as in Eq. 4. The vector $\vec{B}_i^{+}$ denotes the ith row of the inverted basis $\mathbf{B}^+$. By reordering, we obtain

$$c_i = \sum_{j=1}^{m}\sum_{k=1}^{m} \vec{B}_i^{+}\,(\vec{B}_j * \vec{B}_k)\; e_j s_k. \tag{6}$$

This equation can be rewritten as a matrix multiplication, but one of the two coefficient vectors has to be integrated into it. To do so, we define a new matrix $\mathbf{R}$ written in terms of either $\vec{s}$ or $\vec{e}$:

$$\begin{aligned} \vec{c} = \mathbf{R}(\vec{s})\,\vec{e} \;&=\; \mathbf{R}(\vec{e})\,\vec{s}, \\ \text{with}\;\; \mathbf{R}_{ij}(\vec{v}) &= \sum_{k=1}^{m} \vec{B}_i^{+}\,(\vec{B}_j * \vec{B}_k)\; v_k \end{aligned} \tag{7}$$

The $m \times m$ matrix $\mathbf{R}$ carries out any reflectance computation inside the linear subspace. Equation 7 also shows that an arbitrary choice of some basis does not necessarily lead to a diagonalization of $\mathbf{R}$. However, it is at least possible to use specialized hardware to apply this matrix at every reflection event [Peercy et al. 1995]. Current commodity graphics hardware do also allow for an implementation using the GPU.

In the following, we discuss how to modify the basis functions such that *componentwise multiplications* alone, *with diagonalization of* $\mathbf{R}$, will suffice for such computations.

2.3 The RGB Factor Model and the Spectral Factor Model

2.3.1 *Spectral Sharpening.* The linear color model we use here is based on an extension, to spectral bases, of an idea called *Spectral Sharpening* in color constancy algorithms in computer vision [Finlayson et al. 1994].

The factor model for RGB 3-vectors in physics-based vision considers explicitly the role of a camera in color formation. In our case, in place of a camera we must instead use the basis set $\mathbf{B}$. The idea of spectral sharpening is to form camera filter combinations that are more narrowband; and that is just what is also required here, but for the basis instead of the camera.

RGB spectral sharpening proceeds as follows. Suppose we have a 31×3 set of camera sensors $\mathbf{Q}$. Its rows are discretized versions of the spectral sensitivity functions $Q_k(\lambda)$ of each sensor. For illuminant $E(\lambda)$ interacting with surface reflectance function $S(\lambda)$, a physically correct RGB color 3-vector $\vec{r}$ is given by

$$r_k \equiv \int E(\lambda)\, S(\lambda)\, Q_k(\lambda)\, d\lambda, \quad k = 1..3. \tag{8}$$

An approximation is formed by

$$r_k \simeq \sigma_k \epsilon_k / w_k\ , \quad k = 1..3 \tag{9}$$

where σ_k is the surface color under equi-energy white light,

$$\vec{\sigma} = \mathbf{Q}^T \vec{S} \tag{10}$$

and ϵ_k is the color of the illuminant,

$$\vec{\epsilon} = \mathbf{Q}^T \vec{E}. \tag{11}$$

The camera scaling term is

$$\vec{w} = \mathbf{Q}^T\ \vec{1}_{31}, \tag{12}$$

where $\vec{1}_{31}$ is a vector of 31 ones. [Borges 1991] carefully considered this approximation and showed that it is accurate provided illuminants (or surfaces) are 'white enough'. In practice, the light can be relatively non–white and still give fairly accurate results under an RGB factor model. More importantly, for our application, it is clear that if the camera sensors $\mathbf{Q}$ are *narrowband* enough then a factor model will hold. Spectral sharpening provides just this needed band limiting, by a judicious combination of the original, broadband, sensors. The idea is that a 3×3 matrix transform of camera sensors can place more energy for each sensor curve into its appropriate wavelength range. Since these new sensors are more like delta functions, the camera white balance is very simple — thus, sharpening is used in digital cameras (see [Drew and Finlayson 2000] for a discussion of optimization methods for producing such a transform).

Usually, since we have the freedom of defining the intermediate, sharpened color space, we set the L_1 norm of each new camera filter to unity (each column of the new $\mathbf{Q}$ sums to 1). So in this case Eq. 9 simplifies to

$$r_k \simeq \sigma_k \epsilon_k\ , \quad k = 1..3 \tag{13}$$

and spectral sharpening allows us to approximate surface color by a simple componentwise multiplication.

2.3.2 *Spectral Factor Model.* We are interested in multiplying full $\mathbb{R}^{31}$ spectra. The best we can do in an optimal fashion for representing spectra that participate in image formation is to form a principal component basis for the spectral curves [Drew and Funt 1992]. Then, as we have seen, since each spectrum is represented as a sum over basis coefficients this necessarily implies a matrix multiply of current light coefficients times the next interaction spectrum coefficients. However, just as we 'sharpened' the RGBs by a matrix transform and worked in the intermediate space, here we can apply the same idea to the basis set. In a camera, we form combinations of the R, G, and B sensors that are optimally narrowband; here, we form combinations of the basis set vectors.

That is, we pre-'sharpen' the basis by a simple matrix transform and then agree to operate within the sharpened basis for all surface or volume interactions [Drew and Finlayson 2003]. Note that no information is lost by such a transform, and accuracy to within the adopted dimensionality of the underlying finite-dimensional model is maintained.

Then we can represent spectral interactions in terms of the low-dimensional coefficients (typically 5 to 7) and calculate interactions using simple componentwise multiplication of the coefficients. Using the new basis, Eq. 6 is reduced to a simple componentwise multiply involving the coefficients of the current light, e_i, times those for the next interaction surface, s_i:

$$\vec{B}_i^+ \, (\vec{B}_j * \vec{B}_k) \;\simeq\; \delta_{i,j}\delta_{j,k} \tag{14}$$

so that

$$c_i \;\simeq\; e_i \, s_i \;, \quad i = 1..m. \tag{15}$$

A set of sharpened basis vectors approximately obeys this equality [Drew and Finlayson 2002]. Hence we have re-cast the color interaction Eq. 13, with a basis in place of the camera, and spectral coefficients in the place of color.

Since the illumination of (spectral) light color can be achieved by one componentwise multiplication, it can be the final multiply in a sequence of reflection calculations, and hence adding light amounts to a *post-illumination* step. In the last step, descending to 3D RGB color, we simply need a $3 \times m$ matrix multiply for each pixel to create color on the screen, where m is the dimension of 'color' (i.e., coefficients) in the basis coefficient space.

2.3.3 *Accuracy.* One problem with using linear models is that only the spectra from which the basis functions are derived are likely to be represented with good accuracy. For this we make use of principal component analysis (PCA) for a set of given spectra, and the first m significant vectors are taken to span an orthonormal linear subspace for the spectra. Other spectra, which have not been considered during the construction of this basis may be very different from their projections into that space.

Particularly in the case of fluorescent (spiky) lights or sharp cutoff bands, we should make use of a dedicated, or 'specialized', basis. Each illumination step as described in Eq. 7 makes use of a result projected back into the linear subspace and hence at every interaction the linear representation may move farther from an accurate representation of the product spectrum. This problem is especially relevant if we use multiple scattering or spectral volume absorption. The highest accuracy is achieved when only very few illumination calculations are performed. In case of a local illumination model in combination with 'flat' absorption (alpha blending), only one scattering event is considered with no further transmission events. Another technique especially appropriate for linear color models is sub-surface scattering [Hanrahan and Krueger 1993]. This method uses only very few reflections beneath a surface. Yet the spectral absorption (participating medium) is important for realistic results, so using spectra can greatly improve correctness; since there are only relatively few absorption events the accuracy is still acceptable.

3. DESIGNING SPECTRA

The technique described in this section seeks to extend a scene containing real world reflectance and light spectra by creating additional artificial materials and lights that fulfill certain criteria:

(1) a constructed light/reflectance should produce user-chosen colors in combination with given reflectances/lights;
(2) spectra may also be represented in a lower dimensional linear subspace model for which the approximation error should be minimal;
(3) to regularize the solution we minimize the second order difference of the discrete spectrum; this provides smoothness of the solution and improves convergence;
(4) physically plausible spectra should be positive, which enters the optimization as a lower bound constraint.

The first three of these points are expressed as linear least squares problems. This allows us to weight and combine different criteria and to employ standard solution methods.

All settings involved in the design process are represented as a palette of spectra and combination colors as shown in Fig. 1. The display uses columns for lights and rows for reflectances. In the example the lights act as input to the design algorithm while the reflectances were open for redesign. For any light-reflectance combination, the user may define a desired color that should result from the illumination. It is displayed in the framed sub-square of the color patch. Its surrounding area shows the color that the actual resulting spectra produce in combination with each other.

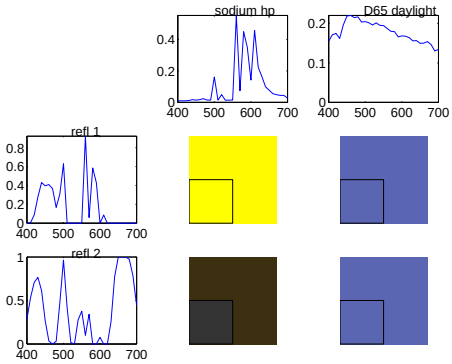


Fig. 1. Spectral design of two material reflectances shown on the left of their representative rows. The colors formed under two different illumination spectra are shown in the squares in the respective columns where D65 (right column) produces a metameric appearance.

The design was successful if the desired and the actual colors are similar enough. The appearance of 'refl 2' under the high-pressure sodium lamp is dark brown instead of the desired gray, which is acceptable in this example.

3.1 Matrix Formulation

It is possible to approach the design problem by solving a linear equation system for a spectrum $\vec{x}$

$$\mathbf{Q}_{rgb,31}\text{diag}(\vec{E})\vec{x} = \vec{c}, \tag{16}$$

where $\mathbf{Q}_{rgb,31}$ is the spectrum to RGB conversion matrix[2]. The solution $\vec{x}$ will be a reflectance producing the user-specified color tri-stimulus $\vec{c}$ under the given illumination spectrum $\vec{E}$. Further, one might ask for multiple lights $\vec{E}_k$ to produce colors $\vec{c}_k$ with reflectance $\vec{x}$. One can solve for this by vertically concatenating the illumination matrices $\mathbf{Q}_{rgb,31}\text{diag}(\vec{E}_k) = \mathbf{Q}_{rgb,31}^{(\vec{E}_k)}$ into a matrix $\mathbf{M}$ and their respective color forcing vectors $\vec{c}_k$ into a vector $\vec{y}$. As there might not be a spectrum that fulfills all conditions exactly we switch from solving an equation system to a quadratic minimization problem:

$$\min_{\vec{x}}||\mathbf{M}\vec{x} - \vec{y}|| = \min_{\vec{x}}\left[\vec{x}^T\mathbf{M}^T\mathbf{M}\vec{x} - 2\vec{y}^T\mathbf{M}\vec{x}\right], \tag{17}$$

An unconstrained solution would be available via the pseudo-inverse $\mathbf{M}^+ = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}$ as $\vec{x} = \mathbf{M}^+\vec{y}$. Alternatively, we use quadratic programming (QP), because it allows the inclusion of lower and upper bound constraints for the components of $\vec{x}$. Note that the entire design could be carried out for a light $\vec{E}$ instead of a reflectance $\vec{S}$, by replacing $\vec{E}$ with $\vec{S}$ in Eq. 16. The solution $\vec{x}$ would then contain a light $\vec{E}$ producing color $\vec{c}$ when illuminating the given reflectance $\vec{S}$. This outlines the main idea behind the design method. We will refine it in the following by adding more optional criteria, such as linear subspace model error minimization and smoothness via minimal second order differences. Finally, all criteria are weighted and combined by concatenating them to form $\mathbf{M}$ and $\vec{y}$ in a single QP problem.

As shown in § 2.2.1, color computation can also be performed in the linear subspace. The $3 \times m$ matrix that takes an m-vector representation in basis $\mathbf{B}$ of Eq. 2 directly to RGB is $\mathbf{Q}_{rgb,m} = \mathbf{Q}_{rgb,31}\,\mathbf{B}$. The least squares problem

[2]The matrix may be formed as $\mathbf{Q}_{rgb,31} = \mathbf{Q}_{rgb,xyz} \cdot \mathbf{Q}_{xyz,31}$, where the rows of $\mathbf{Q}_{xyz,31}$ are the 3×31 set of color matching functions in the CIE XYZ model [Wyszecki and Stiles 1982] and $\mathbf{Q}_{rgb,xyz}$ is a hardware (monitor) dependent 3×3 matrix to transform XYZ to RGB.

that minimizes the error when computing illumination in the subspace is expressed as:

$$\min_{\vec{s}} \|\mathbf{Q}_{rgb,m}\mathbf{R}(\vec{e})\,\vec{s} - \vec{c}\|, \text{ corresponds to}$$

$$\min_{\vec{S}} \|\mathbf{Q}^{(\vec{E})}_{rgb,m,31}\,\vec{S} - \vec{c}\|, \qquad \text{with}$$

$$\mathbf{Q}^{(\vec{E})}_{rgb,m,31} = \mathbf{Q}_{rgb,m}\,\mathbf{R}(\mathbf{Q}_{m,31}\,\vec{E})\,\mathbf{Q}_{m,31}, \tag{18}$$

where $\mathbf{R}(\vec{e})$ is the matrix from Eq. 7 that performs the illumination calculation of light $\vec{E}$ and surface $\vec{S}$ using their $\mathbb{R}^m$ subspace representations $\vec{e} = \mathbf{Q}_{m,31}\vec{E}$ or $\vec{s}$ in analog form. As mentioned in § 2.3, using the spectral factor model, matrix $\mathbf{R}(\vec{e})$ can be replaced by $\mathbf{diag}(\vec{e})$. In order to have a general description we have retained matrix $\mathbf{R}(\vec{e})$ above.

From the previous discussion we can express color objectives for illumination in the point sampled 31-dimensional spectral model and in the m-dimensional subspace model. The following criterion seeks to minimize the difference between the resulting colors from these two illumination methods to be close to a zero 3-vector $\vec{0}_3$:

$$\min_{\vec{S}} \|\mathbf{F}(\vec{E})\,\vec{S} - \vec{0}_3\|, \tag{19}$$

$$\mathbf{F}(\vec{E}) = \mathbf{Q}^{(\vec{E})}_{rgb,31} - \mathbf{Q}^{(\vec{E})}_{rgb,m,31}. \tag{20}$$

The third criterion is smoothness. While the previous two criteria are aimed at accurate color reproduction, this one is introduced to allow control over the general shape of the spectrum and to provide regularization reducing the search space of the optimization. An optimal solution for given design colors with minimum error can lead to spiky spectra with large extrema. A commonly used indicator for roughness of a curve is the integral over the squared second derivative or second order differences in our discretized model. Other indicators are possible, but this one can easily be expressed in the following form:

$$\min_{\vec{S}} \|\mathbf{D}\,\vec{S} - \vec{0}_{31}\|, \text{ with}$$

$$\mathbf{D} = \text{ toeplitz}([\,-1\ \ 2\ \ -1\ \ 0\ \ \cdots\ \ 0\,])\ ,\ \vec{0}_{31} = \text{zero 31-vector.} \tag{21}$$

$\mathbf{D}$ is a tri-diagonal matrix having 3-vector $[-1,\ 2,\ -1]$ on the three middle diagonals and zero everywhere else, which is also called a Toeplitz matrix. The whole matrix $\mathbf{D}$ is normalized by $1/(\sqrt{31}\,\|-1\ 2\ -1\|)$: the $\sqrt{31}$ takes care of the number of rows of the matrix so as not to make smoothness more important than the design color matrices. Those we normalize by $1/\sqrt{3}$ in order to have comparable importance. This is relevant when the residues of all above criteria are combined in a sum of squares, as we will discuss next.

3.2 Combined Optimization Function

Each of the design criteria is expressed as one of the matrices $\mathbf{Q}^{(\vec{X})}_{rgb,31}, \mathbf{Q}^{(\vec{X})}_{rgb,m,31}, \mathbf{F}, \mathbf{D}$ with accompanying objective vectors (target colors or zero vectors). The design matrix $\mathbf{M}$ is formed by vertically concatenating these criteria matrices. Similarly, the associated forcing vectors are stacked to form $\vec{y}$. The different criteria are weighted by $\omega_{\{ij|F|D\}}$ for design colors $\vec{c}_{ij}$, error matrix $\mathbf{F}$, and smoothness $\mathbf{D}$ respectively. These weights provide control over the convergence of the minimization and may all be set to 1. We compute a minimum error solution for an overdetermined system $\mathbf{M}_i\,\vec{x} = \vec{y}_i$ for a surface $\vec{S}_i$ corresponding to the set of stacked equations:

$$\begin{bmatrix} \omega_{i1}\mathbf{Q}^{(\vec{E}_1)}_{rgb,31} \\ \omega_F\omega_{i1}\mathbf{F}(\vec{E}_1) \\ \omega_D\mathbf{D} \end{bmatrix} \cdot \vec{x} = \vec{y} = \begin{bmatrix} \omega_{i1}\vec{c}_{i1} \\ \vec{0}_3 \\ \vec{0}_{31} \end{bmatrix}, \tag{22}$$

We solve this system for a minimum error solution using the form of Eq. 17. The solution $\vec{x}$ will contain the desired reflectance $\vec{S}_i$ producing color $\vec{c}_{i1}$ with light $\vec{E}_1$. If there are several colors that should be produced in combination

with different lights $\vec{E}_j$, the upper two blocks are vertically repeated for each $\vec{E}_j$, since the smoothness criterion $\mathbf{D}$ only needs to be included once. In the following we will consider the simultaneous creation of several spectra.

3.2.1 *Free Metamers.* In the above formulation the design of one spectrum $\vec{S}_i$ is independent of the other spectra that are to be designed. However, it is possible to solve for all needed spectra simultaneously, by combining their individual design matrices $\mathbf{M}_i$ in a direct sum. This means to concatenate the matrices diagonally and to fill the remaining elements with zeros $\emptyset$ in the form $\mathbf{M} = [\mathbf{M}_1\ \emptyset; \emptyset\ \mathbf{M}_2]$, where the semicolon denotes vertical concatenation. The respective forcing vectors $\vec{y}_i$ are stacked as well and the the solution vector $\vec{x}$ will contain several spectra concatenated.

We will use this to include 'free' colors into the design: we create two spectra $\vec{S}_i, \vec{S}_j$ and instead of defining their desired color as part of $\vec{y}$, we will leave the actual color open and retrieve it as part of the solution in $\vec{x}$. This is useful if we want those two reflectances to look the same under a light $\vec{E}_a$, but do not care what specific color they will actually form as long as they are metameric. This can then be combined with further design colors for a different light $\vec{E}_b$. More formally, we solve for a weighted solution $\{\vec{c}_{ia},\ \vec{S}_i,\ \vec{S}_j\}$ of the system

$$\begin{aligned} \mathbf{Q}_{rgb,31}^{(\vec{E}_a)}\,\vec{S}_i &= \mathbf{Q}_{rgb,31}^{(\vec{E}_a)}\,\vec{S}_j, \\ \mathbf{Q}_{rgb,31}^{(\vec{E}_b)}\,\vec{S}_i &= \vec{c}_{ib} \end{aligned} \tag{23}$$

using an illumination matrix as defined after Eq. 16 and with color $\vec{c}_{ib}$ given for a surface $\vec{S}_i$ under light $\vec{E}_b$: i.e., we ask this surface $\vec{S}_i$ under another light $\vec{E}_a$ to have the same color as surface $\vec{S}_j$ under that light. As a stacked matrix Eq. 23 becomes

$$\begin{bmatrix} -\omega_{ia} & 0 & 0 & & \vec{0}_{31} \\ 0 & -\omega_{ia} & 0 & \omega_{ia}\mathbf{Q}_{rgb,31}^{(\vec{E}_a)} & \vec{0}_{31} \\ 0 & 0 & -\omega_{ia} & & \vec{0}_{31} \\ -\omega_{ja} & 0 & 0 & \vec{0}_{31} & \\ 0 & -\omega_{ja} & 0 & \vec{0}_{31} & \omega_{ja}\mathbf{Q}_{rgb,31}^{(\vec{E}_a)} \\ 0 & 0 & -\omega_{ja} & \vec{0}_{31} & \\ 0 & 0 & 0 & & \vec{0}_{31} \\ 0 & 0 & 0 & \omega_{ib}\mathbf{Q}_{rgb,31}^{(\vec{E}_b)} & \vec{0}_{31} \\ 0 & 0 & 0 & & \vec{0}_{31} \end{bmatrix} \cdot \vec{x} = \vec{y} = \begin{bmatrix} \vec{0}_3 \\ \vec{0}_3 \\ \omega_{ib}\vec{c}_{ib} \end{bmatrix}. \tag{24}$$

The involved weights ω can be changed from the default value 1 to steer the importance of this color criterion over others. The resulting $\vec{x}$ contains the 'free' color $\vec{c}_{ia} = \vec{c}_{ja}$ in the first three components, and after that two 31-component vectors for reflectances $\vec{S}_i$ and $\vec{S}_j$. This setup becomes interesting when used with upper and lower bounds on $\vec{x}$, because then the 'free' color can be forced into a given interval without being specified precisely. The blue metameric color under light D65 in Fig. 1 was obtained using this method.

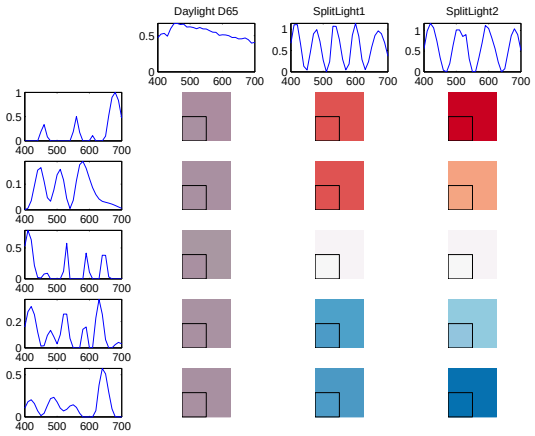
4. EVALUATION AND VISUAL RESULTS

In the following, we will demonstrate the use of our spectral design method in several contexts. We will start with a palette design, followed by an error evaluation for different design conditions. Beyond considering palettes by themselves we will also show their use in practical spectral rendering examples in 3D surface graphics and volume rendering. The set of Matlab scripts for the design method and example setups of this paper along with a Java version of the spectrum generator are available as supplementary material at the URL given at the end of the introduction.

4.1 Example palette design

An example palette design is shown in Fig. 2. The target colors (shown in the framed sub-squares) under the light in the third column were taken from a palette of the map color brewer tool [Harrower and Brewer 2003]. A light fulfilling these colors is generated with our method and is denoted SplitLight 2. The center column colors are chosen to visually merge each of the two red tones in column 3 and separately the two blue tones, replacing them by their average. These colors result when switching to designed illuminant SplitLight 1. In contrast, the first column, illuminated by measured standard daylight D65, is set to a single metameric average color for all five reflectances. The spectra of the two artificial 'split light' sources are initially chosen as shifted positive sine waves of different period lengths ($60nm$ and $85nm$). For the setup the smoothness weight ω_D was set to 1 and subspace error minimization was omitted. All light sources are initially normalized to a luminance $Y = 6$ and the reflectances are allowed to have magnitudes in $[0, 1]$. This scaling factor was determined experimentally through a preliminary unbounded design.

In a first design phase we create reflectances fulfilling the given colors. Here, we choose importance weights $\omega_{i,1} = 4$ for the first column, leaving the remaining weights at 1. This gives correct metamers under daylight D65 and gets the remaining palette colors approximately right. In a second phase we use the newly created reflectances to re-create the two 'split lights' to produce the given colors more exactly. The resulting spectra are shown in the graphs in Fig. 2.




```
d65    = [0.4702 0.5197 0.5304 0.4924 0.5958 0.6645 0.669 0.6525 0.6582 0.6179 0.6213 0.6122 0.5952 0.6117 0.5929
          0.5906 0.5679 0.5469 0.5441 0.5038 0.5111 0.5089 0.4981 0.4731 0.4754 0.4543 0.4555 0.4674 0.4447 0.3958 0.4066];
split1 = [0.6678 1.102 1.107 0.6561 0.1297 0.043 0.4404 0.8899 0.9883 0.7145 0.273 0 0.2365 1.07 1.065
          0.7561 0.292 0.0444 0.1839 0.858 1.136 0.8329 0.288 0.05081 0.2354 0.5776 0.8495 0.959 0.8936 0.6813 0.3666];
split2 = [0.5569 0.9856 1.172 1.064 0.7426 0.3406 0.03923 0 0.2016 0.6381 1.012 1.013 0.8544 0.8999 0.3158
          0 0.01159 0.3446 0.6981 1.119 1.049 0.8205 0.5588 0.2338 0 0.07294 0.4685 0.8727 1.039 0.9004 0.5156];

refl1  = [0 0 0 0 0 0.1908 0.337 0.08154 0 0 0 0 0 0 0
          0.1881 0.5014 0.1729 0 0 0.005627 0.1097 0 0 0 0.09737 0.524 0.8975 1 0.8346 0.4686];
refl2  = [0 0.006213 0.03533 0.0938 0.155 0.1646 0.1112 0.04728 0.0334 0.0771 0.1367 0.1579 0.1164 0.04379 0.005049
          0.03718 0.1136 0.1759 0.1892 0.1624 0.1227 0.08678 0.05953 0.04195 0.03305 0.02917 0.02617 0.02203 0.01681 0.01118 0.005552];
refl3  = [0.5278 0.7863 0.6334 0.2164 0.02129 0.01284 0.07832 0.09049 0 0 0 0 0 0.1197 0.5766 0
          0 0 0 0 0.4177 0.1081 0 0 0 0.3824 0.3816 0.03062 0 0 0 0];
refl4  = [0.158 0.2753 0.3143 0.2531 0.1204 0.01347 0.01566 0.09516 0.1335 0.08491 0.02991 0.09995 0.2566 0.2573 0.07249
          0 0 0 0.141 0.1585 0.000607 0.001365 0.2275 0.3689 0.2555 0.0623 0 0 0.02455 0.04152 0.03065];
refl5  = [0.1036 0.1798 0.2026 0.1579 0.07112 0.01515 0.04265 0.134 0.2162 0.2319 0.1768 0.1033 0.07003 0.08917 0.1286
          0.1434 0.1117 0.04218 0 0 0 0 0 0.07356 0.3759 0.5742 0.5092 0.2921 0.09508 0.003234 0 0];
```

Fig. 2. The reflectance spectra on the left of each row are designed to be metameric under daylight (colors column 1) and to gradually split off into 3 and 5 distinguishable colors under two artificial 'split light' sources. The resulting reflectance spectra are given below the figure.

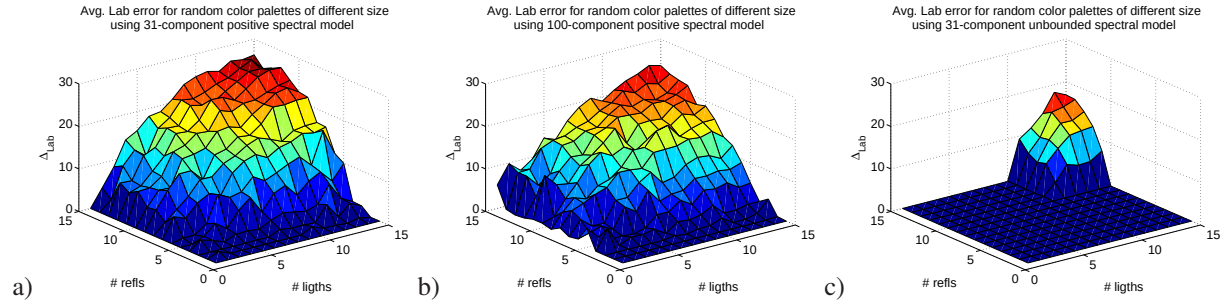


Fig. 3. Each graph shows the average L*a*b* error in the design process for palettes of given sizes, constraining all light-reflectance combination colors for several palettes of different sizes. Changing spectral models and constraints results in different design error: a) the positivity constrained 31D model, b) the positivity constrained 100D color model, c) 31D without positivity constraint.

4.2 Design Error with respect to number of constraints

Expressing the design criteria as soft constraints allows us to always obtain a solution, but possibly with errors depending on how well the criteria agree with each other. In order to obtain a better understanding of these errors we performed a number of automated tests on palettes of varying sizes. For each of the tests (with errors displayed in Fig. 3), we are requesting fixed random combination colors between each reflectance and light (uniformly distributed in RGB space). Lights and reflectance are formed by first creating reflectances for fixed random lights and then recomputing lights for the new reflectances. The bottom left and right axes of each graph in Fig. 3 indicate the numbers of reflectances and lights, respectively. The vertical axis denotes the average L*a*b* distance between designed and actual light-reflectance combination colors over the entire palette of a given size. While an error $\Delta_{Lab} < 3$ is visually indistinguishable, we found that errors up to 10 are still acceptable. The 31-dimensional positive spectral model of Fig. 3a lies in this acceptable error range for palette sizes of up to 7×7. Each RGB combination color adds 3 constraints, which for 10 colors are matched by the degrees of freedom of a 31-dimensional spectrum to be designed. Thus, without positivity constraint an error is expected to occur after a palette size of 10×10 as observable in Fig. 3c.

Methods to reduce the error are to increase the dimensionality of the color model (Fig. 3b) or to drop the positivity constraint (Fig. 3c). The drastic reduction in error shows that positivity imposes a major restriction on the design process. Reducing the weight of the smoothness term ω_D has a similar error decreasing effect as increasing the dimensionality of the color model since both are different ways to regularize the solution. For our experiments we have kept a fixed $\omega_D = 0.001$.

4.2.1 *Preserving distance between colors.* When setting up several colors a designer sometimes need not closely specify just what color is actually produced, but rather that the colors of two objects should be very similar or notably different. This idea is the motivation for our second type of evaluation. Here, we do not look at the preservation of the actual color in the design, but rather the distances between them. In an $N \times M$ palette setup, we consider each of the $\frac{1}{2}(N \times M)^2$ color pairs, excluding duplicate pairings.

In particular, we want to see how well the (perceptual) distance between the desired colors matches the (perceptual) distance between the actual colors produced by the designed spectra. Similar to the previous analysis, the evaluation in Fig. 4 shows that positivity is a rather strong constraint, but that increasing the dimensionality of the underlying spectral model can be used to compensate for it.

4.3 Spectral Surface Graphics

To implement spectral raytracing we have used the physically based rendering toolkit (PBRT) [Pharr and Humphreys 2004]. Its modular design allows us to easily extend the color model to a 31 component linear model with the appro-

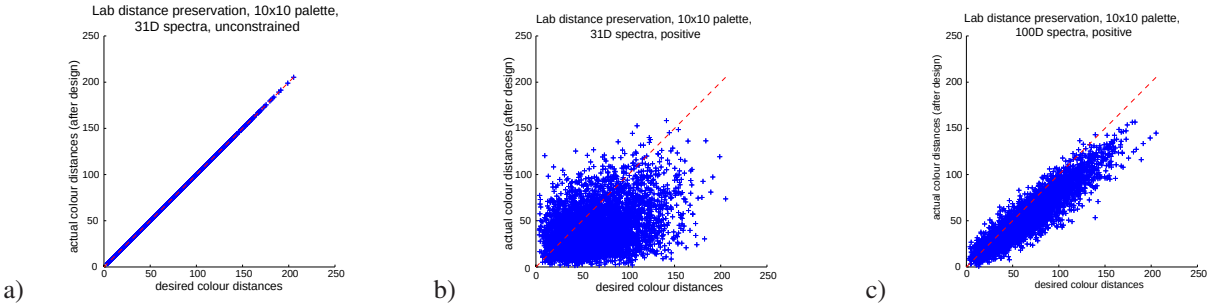


Fig. 4. Preservation of color distances for a 10×10 palette size. Each point in the graphs represents a certain pair of color entries in the palette. Its position along the horizontal axis indicates the L*a*b* distance between the desired colors and the vertical position indicates distance of the resulting color pair after the design. A position close to the diagonal indicates how well the distance within a pair was preserved in the design. a) 31-D spectra, unconstrained; b) and c) positivity constrained spectra with 31 and 100 dimensions, respectively.

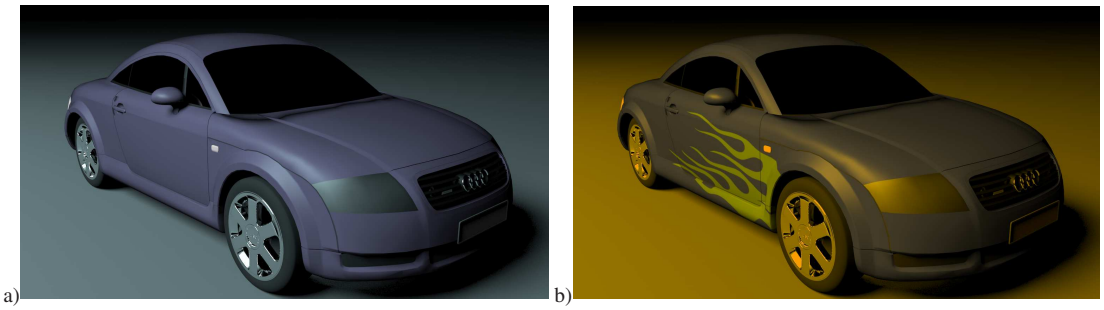


Fig. 5. Car model rendered with PBRT. (a) The spectral materials used in the texture are metameric under daylight D65, resulting in a monochrome appearance. (b) Changing the illumination spectrum to that of a high pressure sodium lamp, as used in street lighting, breaks apart the metamerism and reveals additional visual information.

priate color space transformation matrices. Also, we have added support to load spectral textures, and a Python script to replace RGB values by linear combinations of red, green, and blue spectra to facilitate rendering of conventional RGB scenes with the modified spectral renderer. Fig. 5 shows two impressions of the same scene under different illumination. Lighting is provided by an area light source emitting one of the two illumination spectra of Fig. 1. The spectral texture for the flame job on the side of the car uses the reflectances from the same palette. Thus, the daylight metamerism makes the texture disappear, while the nighttime sodium street lamp illumination breaks apart the metamerism and makes texture details appear. Since here we needed the metamers to match exactly, we set their importance weight to 10, while leaving the weighting for the colors under the sodium lamp at 1.

4.4 Rendering Volumes Interactively

To further illustrate the efficacy of the design method and the palette tool, we consider two volumetric data sets: an engine and a frog. Each is represented by a volumetric grid of scalar density values. Optical properties, such as opacity and spectral reflectance are assigned to different densities via a spectral transfer function. For each data set a specific palette is produced that contains light spectra and reflectances that are assigned to distinct parts of the data (ranges of density values) via the transfer function. The volume is rendered via a *post-illumination* step: the images are first rendered with a flat white light, or rather, lighting having all-unity basis coefficients. Then lighting is changed. The actual raycasting is performed once for a given viewpoint and all subsequent images for changing light can be

computed in real time by simply multiplying the reflected coefficients for spectra from flat white illumination recorded in an image pixel by the new light's basis coefficients. For further details on this method along with a user-friendly widget to control different lighting see [Bergner et al. 2005].

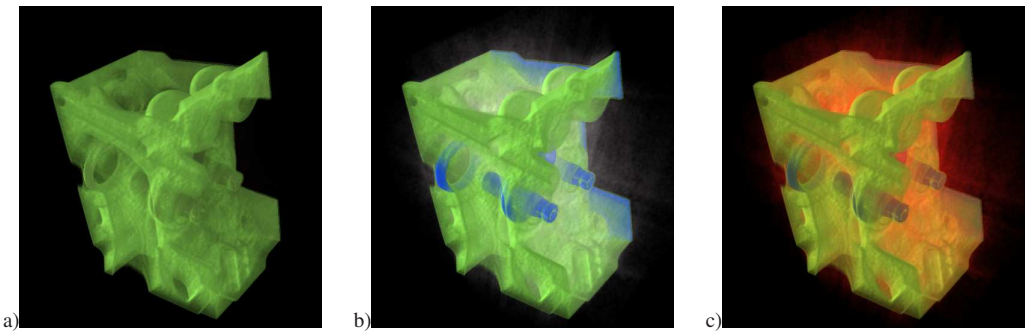


Fig. 6. Engine block rendered using metamers and color constancy. The three images in the figure are re-illuminated without repeating the raycasting.

In Fig. 6(a) an engine block and inner parts are metameric; 'smoke' (reconstruction noise) is present, but is colored black via the metameric black mechanism. Thus, it is invisible. Fig. 6(b) has the same reflectances, but under a different light designed such that inner parts are now distinguishable from the engine block, which itself has kept a constant color from the previous light. The 'smoke' appears white now. Under another light, in Fig. 6(c), the smoke changes color to be red, whereas the other two parts keep their appearance as much as this is possible in the context of having surrounding smoke be of different color.

Another example, a frog, is shown in Fig. 7. Again, the first image shows all materials as being metameric. Notably, this color was not chosen directly, but instead arises as 'free color', as described in § 3 — the design process has chosen the color, on the basis of the optimization. In Fig. 7(b), a new light is gradually mixed in such that the metamers begin to break apart. Additionally, that new light makes the body of the frog go black so that it gradually disappears. In Fig. 7(c), the light dominant in (a) has been switched off. As a result, the body turns dark and the inside structures remain in a distinct color.

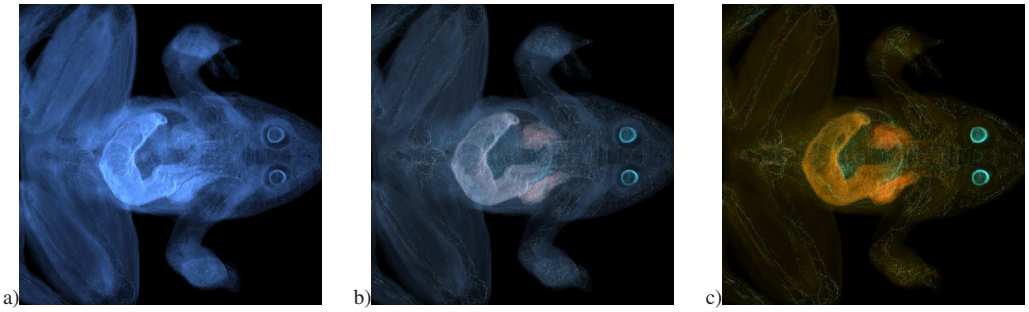


Fig. 7. Frog rendered with four different materials at different mixtures of light spectra.

5. DISCUSSION

In computer graphics the step from RGB to full spectral color models for illumination calculations is typically made to increase realism of the renditions, since the increased dimensionality of a spectral light and reflectance model allows for more accurate illumination computations. At the same time it leaves the creator of a spectral scene with the tasks of specifying such additional degrees of freedom of the appearance to the best effect. Using real world measurements would of course be the method of choice to attain highest realism. A different direction is to use the additional freedom to include effects that are specific to spectra. Since the color appearance of a material is dependent on the illumination and typically differs under varying illumination, the key idea is to conduct a combined design for pairs of reflectances and lights.

The example palette in Fig. 1 shows a simple setup, using metamerism under one light, that breaks apart into distinguishable colors under a different light. This palette is used in a spectral texture of Fig. 5 to illustrate how non-/metamerism under specific lighting can be used to reveal texture details selectively via lighting change. A possible extension to such a setup would be a dynamic scene in which the user introduces the metamer-breaking light via a flashlight. In that case, the flashlight would become a metaphor for revealing additional detail in the scene.

The same idea of merging and splitting metamers may be applied hierarchically to a palette, in Fig. 2. The idea here is that under one illumination (daylight) the entire scene has a homogeneous appearance. Under additional 'split' lights there are 3 and then 5 classes distinguishable leading to different levels of discernability, e.g., in a rendition of a map. Using this in a spectral texture similar to Fig. 5 forms a new way to scale visual complexity by selectively introducing additional texture detail.

Also, controlling the lighting in the spectrum design addresses a problem pointed out in [Smits 1999] in that some colors can only be modeled with physically implausible reflectances that have magnitudes larger than 1. However, when considering color as perceived from an image one does not deal with reflectances, but rather reflected spectra, which involve the multiplication of a light source spectrum. By considering colors for light-reflectance combinations, the [0,1] bounds for reflectances can be maintained by scaling up the intensity of the light sources. One way to obtain a suitable scaling is to initially produce reflectances without upper bound. If their maximum magnitude is above 1, we may scale up all lights by that magnitude value and repeat the design with a forced upper bound of 1 for the reflectances.

The design errors discussed in § 4.2 were shown to depend on the number of constraints and also on enforcement of positivity. In response to this situation, the user has to make a decision. Firstly, the user could adjust the importance weights to drive the compromise of conflicting criteria, e.g., favoring the color combinations that have the highest visible errors or that matter most to the final result. This in particular applies to the metamers, that are supposed to look exactly the same. To achieve this we set the importance of the metameric color patches 4 or 10 times higher than that of the remaining ones. Another solution are 'free' colors, since these allow us to choose a 'natural' metameric color that is easiest to fulfill in concert with the other given color conditions.

As an additional option, in case precise color fulfillment is very important, the positivity constraint could be dropped to allow spectra with negative components. As a result, the optimization may more likely fulfill the given criteria without any error (see Fig. 3c). The same figure also indicates that choosing an even higher dimensional spectral model is yet another possibility for satisfying more constraints.

This kind of iterative design process of adjusting criteria to what is possible has not been performed in our automatic evaluation, but is feasible in practice. An interactive 'live' update feature of our Java implementation was found to be helpful in this regard. It performs the re-design (constructing and solving the criteria matrix) in a background thread whenever changes are made. This immediate feedback allows the user to adjust conflicting colors or constraints the moment they are introduced.

5.1 Future directions

The design has so far only considered colors of single illuminants combined with single reflectances. In rendering practice, one may also be interested in mixtures of different illuminants as well as materials of combined reflectances. While the light source mixture would be linear due to additive superposition of the electromagnetic field, the same

may not hold for reflectances of mixed materials.

More advanced illumination models could be taken into account. For instance a bi-reflectivity matrix relating illuminating wavelengths to re-emitted wavelengths [Devlin et al. 2002] could be used to model fluorescence. In fact, one could readily apply such a matrix, replacing $\mathbf{diag}(\vec{E})$ in Eq. 16 and design a light $\vec{E}$ to produce a given color on a fluorescent surface.

A feature similar to the free color approach that creates two or more colors that are as distinct as possible rather than equal could be useful. In that case, it might be helpful to switch to a more perceptually uniform color space. A linear approximation to CIELAB [Sharma and H.J. Trussell 1997] could be of use here. Work in such a direction should also consider perceptual issues of palette creation [Harrower and Brewer 2003].

Just like physical simulation can replace manual geometrical modeling, e.g., for cloth or water surfaces employing spectral rendering could achieve the same for lighting and color appearance. That is, with sets of materials designed to produce a well balanced and distinct appearance under various lights. Instead of changing the palette to create a different mood the change would be implicit under a different light.

With a proper paint mixture model one could look for feasible spectra that could be mixed from paints or in combination with other base materials, such as metals or textiles or even skin as in the case of face masks and make-up.

All our examples are given in the context of image synthesis. But a method to construct spectra might also be used for reconstruction of spectra in a computer vision setting, e.g., using multiple images of the same scene taken under different color filters or different controlled illumination. Here, each color filter would give rise to a distinct spectrum to camera RGB transformation matrix that takes the effect of the filter into account via a pre-multiplied diagonal matrix containing the filter absorptance spectrum. The optimization of the design method could then reconstruct a spectrum that simultaneously fulfills the different recorded RGB values of a given image pixel under the respective color filtered transformation matrices.

6. CONCLUSIONS

Even though spectral rendering is fairly easy to implement, it is still not widely used. Performance issues are only partly a reason. In fact, due to cache coherency and with cheap component-wise illumination calculations, the drop in performance is not significant. Rather, we felt that it is the lack of spectral materials and the difficulty of reusing existing RGB setups and textures within a spectral setup that have posed an obstacle to users.

To close this crucial gap in the design pipeline, we have devised a spectral palette tool that allows the user to create a set of lights and reflectances such that different parts of a rendering can be enhanced, or be made to disappear, in real time and interactively. We formulated the design process as a least squares problem, yielding global minimization of the criteria. The design scheme and optimization is novel in both graphics as well as in color science. The resulting set of spectra and colors have utility in the visualization of volume data, but their usefulness is not restricted to this arena or to surface graphics. In fact, with the liberty to inject actual physical spectra for any of the components, one may design appropriate lights to attain specific perceived colors when viewing real physical subjects.

REFERENCES

ABDUL-RAHMAN, A. AND CHEN, M. 2005. Spectral volume rendering based on the Kubelka-Munk theory. *Computer Graphics Forum 24,* 3, 413–422.

ANDERSON, M., MOTTA, R., CHANDRASEKARAND, S., AND STOKES, M. 1996. Proposal for a standard default color space for the internet — sRGB. In *Fourth Color Imaging Conference: Color, Science, Systems and Applications.* (Nov). Society for Imaging Science & Technology (IS&T)/Society for Information Display (SID) joint conference, Scottsdale, Arizona, 238–245.

BERGNER, S., MÖLLER, T., DREW, M. S., AND FINLAYSON, G. D. 2002. Interactive spectral volume rendering. In *Proc. of IEEE Visualization*. IEEE, Boston, MA, 101–108.

BERGNER, S., MÖLLER, T., TORY, M., AND DREW, M. 2005. A practical approach to spectral volume rendering. *IEEE Transactions on Visualization and Computer Graphics 11*, 207–216.

BORGES, C. 1991. Trichromatic approximation method for surface illumination. *J. Opt. Soc. Am. A 8*, 1319–1323.

DEVLIN, K., CHALMERS, A., WILKIE, A., AND PURGATHOFER, W. 2002. Star: Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics 2002*, D. Fellner and R. Scopignio, Eds. The Eurographics Association, 101–123.

DONNER, C. AND JENSEN, H. W. 2006. A spectral shading model for human skin. In *Rendering Techniques (Proc. of the Eurographics Symp. on Rendering)*. 409–417.

DREW, M. S. AND FINLAYSON, G. D. 2000. Spectral sharpening with positivity. *J. Opt. Soc. Am. A 17*, 1361–1370.

DREW, M. S. AND FINLAYSON, G. D. 2002. Multispectral processing without spectra. Tech. Rep. SFU-CMPT-03/02-TR2002-02, Simon Fraser University School of Computing Science. March. Also: Representation of colour in a colour display system, UK Patent Application No. 0206916.9. Under review, British Patent Office.

DREW, M. S. AND FINLAYSON, G. D. 2003. Multispectral processing without spectra. *J. Opt. Soc. Am. A 20,* 7 (July), 1181–1193.

DREW, M. S. AND FUNT, B. V. 1992. Natural metamers. *Computer Vision Graphics and Image Processing: Image Understanding 56*, 139–151.

FARRELL, J., CUPITT, J., SAUNDERS, D., AND WANDELL, B. 1999. Estimating spectral reflectances of digital artwork. In *Chiba Conference of Multispectral Imaging*.

FINLAYSON, G. D., DREW, M. S., AND FUNT, B. V. 1994. Spectral sharpening: sensor transformations for improved color constancy. *J. Opt. Soc. Am. A 11,* 5 (May), 1553–1563.

GLASSNER, A. 1989. How to derive a spectrum from an RGB triplet. *Computer Graphics and Applications, IEEE 9,* 4 (July), 95–99.

GONDEK, J., MEYER, G., AND NEWMAN, J. 1994. Wavelength dependent reflectance functions. In *Proc. SIGGRAPH 1994*. ACM, Orlando, FL, 213–220.

HANRAHAN, P. AND KRUEGER, W. 1993. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of ACM SIGGRAPH 1993*. Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, 165–174.

HARROWER, M. A. AND BREWER, C. A. 2003. Colorbrewer.org: An online tool for selecting color schemes for maps. *The Cartographic Journal 40,* 1, 27–37. `http://www.colorbrewer.org`.

JOHNSON, G. AND FAIRCHILD, M. 1999. Full-spectral color calculations in realistic image synthesis. *Computer Graphics and Applications 19,* 4: July/August, 47–53.

MARIMONT, D. H. AND WANDELL, B. A. 1992. Linear models of surface and illuminant spectra. *J. Opt. Soc. Am. A 11*, 1905–1913.

MATUSIK, W., PFISTER, H., BRAND, M., AND MCMILLAN, L. 2003. A data-driven reflectance model. *ACM Transactions on Graphics 22,* 3 (July), 759–769.

NOORDMANS, H. J., VAN DER VOORT, H. T., AND SMEULDERS, A. W. 2000. Spectral volume rendering. In *IEEE Transactions on Visualization and Computer Graphics*. IEEE Computer Society, 196–207.

PEERCY, M. S. 1993. Linear color representations for full spectral rendering. In *Computer Graphics (SIGGRAPH '93)*. ACM, 191–198.

PEERCY, M. S., ZHU, B. M., AND BAUM, D. R. 1995. Interactive full spectral rendering. In *Proceedings of the 1995 Symposium on Interactive 3D graphics*. ACM Press, 67–68.

PHARR, M. AND HUMPHREYS, G. 2004. *Physically based rendering*. Elsevier, San Francisco, CA.

SHARMA, G. AND H.J. TRUSSELL, H. 1997. Figures of merit for color scanners. *IEEE Trans. Image Proc. 6,* 7 (July), 990–1001.

SMITS, B. 1999. An RGB to spectrum conversion for reflectances. *Journal of Graphics Tools 4,* 4, 11–22.

STAM, J. 1999. Diffraction shaders. In *Proceedings of SIGGRAPH 99*. 101–110.

STRENGERT, M., KLEIN, T., BOTCHEN, R., STEGMAIER, S., CHEN, M., AND ERTL, T. 2006. Spectral volume rendering using GPU-based raycasting. *The Visual Computer 22,* 8.

SUN, Y.; FRACCHIA, F. C. T. D. M. Jul/Aug 1999. Deriving spectra from colors and rendering light interference. *Computer Graphics and Applications, IEEE 19,* 4, 61–67.

SUN, Y., FRACCHIA, F. D., AND DREW, M. S. 2000. Rendering diamonds. In *Proc. Western Computer Graphics Symposium 2000*. 9–15.

WANG, Q., XY, H., AND SUN, Y. 2004. Practical construction of reflectances for spectral rendering. In *Winter School of Computer Graphics 2004 Posters proceedings*.

WYSZECKI, G. AND STILES, W. 1982. *Color Science: Concepts and Methods, Quantitative Data and Formulas*, 2nd ed. Wiley, New York.

Received Month Year; revised Month Year; accepted Month Year