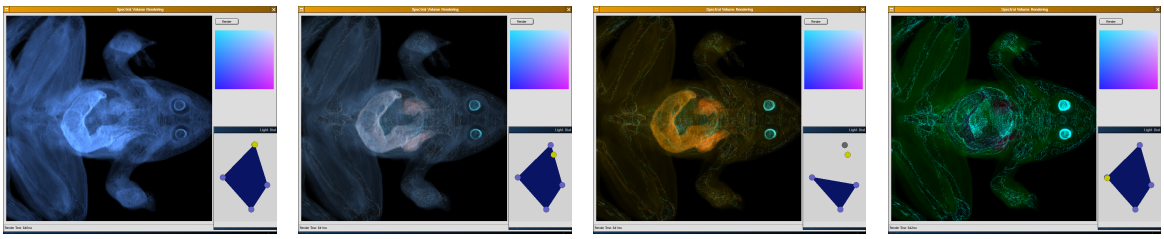
Interactive Spectral Volume Rendering

Steven Bergner* Torsten Möller† Mark S. Drew† Graham D. Finlayson‡

*Department of Simulation and Graphics
University of Magdeburg

†Graphics, Usability, and Visualization Lab
Simon Fraser University

‡School of Information Systems,
University of East Anglia



(a) (b) (c) (d)

Figure 1: Raycast image of a frog using spectral colours. The images can be re-illuminated with different light sources in real-time. By using specially designed materials, this allows one to accentuate or hide additional details.

ABSTRACT

We describe a method for volume rendering using a spectral representation of colour instead of the traditional RGB model. It is shown how to use this framework for a novel exploration of datasets through enhanced transfer function design. Furthermore, our framework is extended to allow real-time re-lighting of the scene created with any rendering method. The technique of post-illumination is introduced to generate new spectral images for arbitrary light colours in real-time. Also a tool is described to design a palette of lights and materials having certain properties such as selective metamerism or colour constancy. Applied to spectral transfer functions, different light colours can accentuate or hide specific qualities of the data. In connection with post-illumination this provides a new degree of freedom for guided exploration of volumetric data, which cannot be achieved using the RGB model.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

Keywords: spectral volume rendering, post-illumination, interactive re-lighting

1 INTRODUCTION

The challenges of volumetric visualization arise from the size of volume data as well as the complexity of the data. One of the key goals of volume graphics is the interactive exploration of volumetric data. Exploration of data includes two important aspects — the change of the viewpoint as well as the change of the visible parts of the data (manipulated through the transfer function.)

*sbergner@cs.uni-magdeburg.de
†{torsten,mark}@cs.sfu.ca, http://gruvi.cs.sfu.ca/
‡graham@sys.uea.ac.uk, http://www.sys.uea.ac.uk/people/graham

IEEE Visualization 2002 Oct. 27 - Nov. 1, 2002, Boston, MA, USA
0-7803-7498-3/02/$17.00 © 2002 IEEE

Much research has focused on the quick rendering of the data in order to enable (real-time) interaction and hence to improve the 3D understanding of the data. The interactive manipulation of the viewpoint allows us to recover a 3D understanding from a series of 2D screen projections making use of the *kinetic depth effect*.

Another important aspect of volume data exploration is the ability to manipulate colour and opacity of voxels to adjust their visibility and appearance. *Transfer function design* is a vibrant research area for the development of new ideas and alternative approaches. The basis of transfer function design has been the assignment of R, G, B, and a transparency value α for each voxel.

Many algorithms gain speed by assuming that the transfer function is fixed. While this seems a rather restrictive assumption, we demonstrate in this paper that we can still manipulate the visualization to potentially gain more insight into our dataset simply by controlling the light sources. We show how to do this in real-time for any volume rendering algorithm. Previous transfer function designs have been carried out using the restrictive RGB colour model. Several researchers have shown that this colour model is not adequate for accurate and high-quality visualizations (see, e.g., [9]). Hence we extend transfer function design to non-restrictive colour spectra. This enables us to exploit to our advantage certain phenomena of colour that are usually undesired. Our main idea is to make use of the concept of *metamers*. Metamers are spectra that appear perceptually identical under a certain light. If we are assigning metameric colours to certain data intensity ranges (using *spectral transfer functions*), we have the ability to visually merge these ranges into one colour under a specific light. By changing the light source we then can visually separate these ranges. This has the potential to help the user explore the relationships between certain intensity ranges.

Spectrally accurate rendering is dependent on a new method [5] for spectral multiplication. In this method, full-spectrum accuracy is maintained throughout light-interaction accumulation by means of a special intermediate low-dimensional colour-vector space, typically of dimension 5 to 7. Furthermore we show that we can change the light source interactively (15 fps) using what we term *post-illumination*. This makes this concept very attractive as an enhanced data exploration technique as well as practical for inclusion into existing visualization packages.

The rest of the paper is structured into the following sections.

Section 2 summarizes previous work. In §3 we present the heart of our paper: the enhancement to transfer function design that enables spectral colours. In §4 we explain our post-illumination framework — this enables the interactive manipulation of rendered images. Section 5 contains necessary details for the replication of our work, while §6 discusses our results. Some ideas for future extensions of this work are discussed in §7, and conclusions are drawn in §8.

2 Related Work

In the past five years we have seen great advances in the area of real-time volume graphics. Special hardware for volume rendering, known as VolumePro [27], was successfully introduced and is commercially available. Much improved consumer hardware is available that has been efficiently exploited for volume rendering using texture mapping capabilities (such as [3, 28]) as well as many new features of modern graphics cards [11]. Exciting new insights into software-based algorithms have brought high-quality interactive visualizations within reach. The UltraVis system [14] achieves 5-10 frames per second on moderate datasets through an optimized memory architecture. Other improved data structures such as an adjacency list or an RLE based data structure were introduced to speed up the splatting algorithm tremendously [24, 11]. The Shear-Warp approach [15] as well as Fourier Volume Rendering [18, 30] enable software-only real-time rendering under quality degradation.

In the area of data exploration through transfer function manipulation, the visualization community has seen great advances as well. Traditionally, a transfer function simply depends on the data values only. In order to strengthen transition regions (regions in which data values change significantly), Levoy[16] suggested considering the data value as well as the magnitude of the local derivative. Kindlmann [12] et al. extended this idea to include the second derivative as well. The contour spectrum, as proposed by Bajaj et al. [1] uses more general criteria such as contour area and gradient integral to select a transfer function.

Enhanced histograms (Laplacian-weighted) have been used as a backdrop to the transfer function widget by Pekar et al. [26] This gives the user a better context for finding and selecting meaningful transfer functions. Kniss et al. [13] have designed a transfer function widget that builds on the ideas of Kindlmann et al. [12] of exploring the 3D transfer function space. They use a probe in order to correlate the real 3D data with the 3D transfer function space.

A totally different approach was taken by Marks et al. [19], who used genetic algorithms to visually explore the space of all possible transfer functions. They created a widget that groups images according to their visual similarities and lets the user pan and zoom within that space in order to select the appropriate visualization.

Ebert et al. [6] automatically determine colour and opacity values by using photographic volumes, such as provided by the Visible Human project. Utilizing additional photographic information is an important step towards rendering photo-realistic volumetric scenes. Thus, their method could be a promising basis for spectral rendering.

Since we are interested in maintaining correct spectral information, we should consider past attempts to accurately model spectra in an efficient manner. Certainly, the most straightforward method of dealing with spectral information is to simply carry spectrally point-sampled data through any rendering calculation. Simply sampling at a uniform interval, however, would mean carrying too many samples — real surfaces and illuminants are typically measured at $>$100 sample points and represented using 31 samples from 400 to 700nm at 10nm intervals. Non-uniform sampling has been proposed to optimize the representation with respect to human perception, and other methods of filtered sampling have been used for most computations. Point sampling has the benefit of remaining most faithful to the original measurement representation, but to achieve high accuracy a large number of samples has to be taken. To reduce dimensionality Rougeron et al. [29] proposed a multiresolution representation for spectra using a wavelet basis. As light spectra bounce through the scene the representation is progressively refined to maintain given error bounds. This model provides controllable quality with a reasonable demand on storage. The only drawback is the computational effort of representing and operating on different resolutions. In volume rendering adaptive refinement is mostly not necessary because most techniques only take primary scattering events into account.

Polynomial models have also been proposed for spectral representation and recently used in image synthesis. However, when the polynomial degree becomes large ($\geq$7) computational instability arises due to the high-power terms. Other linear models have typically been based on eigenfunction expansions of collections of spectral power distributions. Such finite-dimensional models offer optimal representation, in terms of variance-accounted-for, and generally have low dimension (6 to 8). Peercy [25] made good use of such models in an attempt to integrate spectra into surface graphics. However, one is still left with a matrix multiply at each interaction with matter and this presents a stumbling block for using the method .

This issue is precisely what the new colour-vector representation we present here avoids: spectral multiplication is reduced to a simple extension of RGB-colour componentwise multiplication [5]. Of course, in the end what we render is colour, not spectra, using some 3-dimensional colour space such as RGB, CIE XYZ, or CIE Lab. The method we present here essentially creates a new slightly higher-dimensional colour space (5 to 7 component) such that the usual graphics method of multiplying portions of RGBs at each interaction carries through, but with spectral accuracy as well. Three-dimensional colour is calculated only at the last step, after all light interactions are complete.

Finally, Noordmans et al. [23] integrate spectral colour models into a volume rendering pipeline. They discuss different renderers by distinguishing achromatic, elastic and inelastic scattering (particles) independent of achromatic or chromatic absorption (medium). To speed up their rendering, they go from rendering full/actual spectra towards rendering coefficients of materials, which occupy distinct spectral bands. By limiting spectral absorption to be constant for these groups, the rendering computation becomes equivalent to RGBA compositing. The important differences are that these materials get correct colours and that it is possible to have selective absorption by other materials. The use of sharp banded, piecewise constant absorption allows them to set up a general spectral renderer with the ability to show inner structures through solid exterior. The difference in our approach is to gain performance by using an optimized linear colour model throughout the entire rendering process. This allows us to perform full spectral multiplications with minimal computational effort. The rendering is not restricted to a given number of materials, as for Noordmans. Our method also provides more freedom in the way spectra are changing while interacting within the scene. This is essential for more advanced illumination models. As well, we discuss the question of how to sensibly create and use spectra, an important issue for volume rendering and for computer graphics in general not addressed by Noordmans.

3 Interaction Icon

A brute-force extension to current transfer functions would be to allow the user to continuously vary each spectral band individually for each data-point, just as is done with the RGBα model. However, the manipulation of just 3 colour-bands for RGB is found to be unintuitive, and this will get even worse for more colour-bands. Hence our goal is to create certain materials, which can be used to great effect. While material assignment has been used success-

fully in volume rendering before ([4]), using full spectra provides a good deal more freedom in designing transfer functions. A careful strategy is required to sensibly make use of this freedom. A user-oriented integration of spectral concepts into a volume rendering pipeline will have to consider the following steps:

1. Materials and light sources have to be *designed* with an eye to their use in volume rendering: metamers can and should be used to pick out particular regions of interest.

2. Materials are assigned to spectral transfer function values in order to create the most distinct perceptual discrimination.

3. Finally, we are able to navigate through different visualizations by controlling the light source. Since multiple light dimensions are involved we create a new interaction icon.

3.1 Material Design

The goal is to construct colour constellations that are useful for the visualization task. Real-world materials and light sources can form a basis from which to proceed, and in fact it may be useful to model real materials. However, we may also introduce new artificial materials (with non-physical spectral bands, e.g., negative wavelength amplitudes) with differing properties that produce colours that are indistinguishable from real ones, or contrasting colours.

While there is a vast number of possible material designs, we wanted to find materials that would help the user to gain a better understanding of the dataset. Further we were hoping to exploit certain features that colour spaces have to offer. Specifically we made use of the following concepts:

- Metamerism — the colour-matching of differing spectra under one or several lights — can be used to merge volume structures with the background. Alternatively, changing the light can separate materials that were previously metameric, thus creating the ability to emphasize certain structures.

- "Colour constancy" can be used: i.e., given the high dimensionality of spectra we can design materials that move together (or do not move) in colour space. This allows a situation in which only one thing changes at a time, thus focusing attention to a specific feature.

By assigning distinct reflectances to distinct materials we are able to separate materials. However, we created materials such that they interact with the light source in such a way that the resulting colour is identical for our visual sensors (metamers) and hence we have the ability to merge certain materials and therefore focus or de-focus on said materials. The use of concepts of colour constancy allows us to create light sources in connection with the materials, which will leave the colour of some materials the same and only change the colour of the materials we intend to influence with that light.

Hence within the same visualization (without another change of the transfer function) we are able to emphasize and de-emphasize particular regions of interest in connection with particular light sources, independently of other materials in the scene. Therefore we can help the user understand the relationships between these materials. We can create light-sliders that are attached to particular regions and hence will only influence that region. This is demonstrated in Figure 1. In Figure 1a we designed light 1 to make all materials appear equal. Light 2 has been designed to highlight the stomach and intestines of the frog (seen in Figure 1c). Figure 1b shows an in-between frame, where the stomach starts to become more visible.

Since traditional colour theory usually describes only phenomena that occur in the real world, this restricts us to non-artificial metameric materials. However, it would be desirable to design materials that disappear entirely from the image, if so desired by the user. These materials are called *metameric blacks*, i.e., materials with zero RGB under some specific set of lights. This phenomenon doesn't normally appear in real life and hence needed an extension of the theory. In rendering, alpha compositing still takes place, so occlusion of other materials is not in fact undone. Nevertheless, it is possible to introduce materials that only scatter light but do not actually have an alpha value to diminish light that comes from the back. If those X-ray materials turn black under a certain light they entirely disappear from the image sum and free the view to things that they obscure. Since they will always have an X-ray like look, it is not possible to draw crisp surfaces with such materials. In Figure 1d we see that part of the frog's flesh has disappeared. This is due to metameric blacks.

The mathematical details for the material design are published in our technical report [2].

3.2 Spectral Transfer Functions

Transfer functions shall now consist of mappings from data values to spectral materials instead of RGB values. The resulting freedom in design requires a systematic approach to generate both materials and lights, with guiding principles set out in §3.1. The transfer function is designed with a view to assigning spectral materials to distinct features in order to prepare a set of different visual situations for the user. Each combination of lights and materials is meant to highlight distinct qualities of the dataset, with mixing properties of views adhering to mixtures of lights: we may wish to view several features at once, and this can be accomplished by mixing lights that pick out each feature separately.

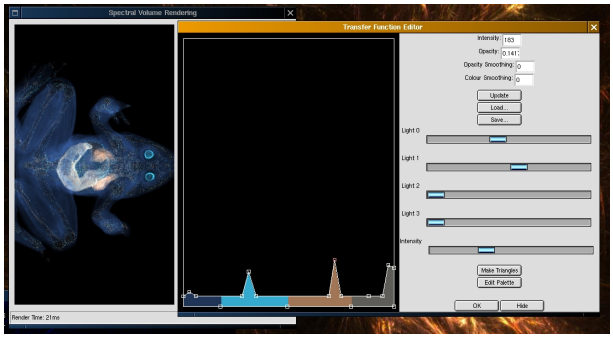


Figure 2: GUI for defining a spectral transfer function.

The transfer function shown in Fig. 2 assigns materials to different intervals of data values that represent distinct segments. The height of the function represents alpha values, which are more opaque for some intervals. The actual colour shown on the screen is determined using a mixture of lights. The colours change as the light for the scene is changed using the control sliders on the right. Thus, the materials can emphasize certain qualities of the dataset. The goal is to have relevant structures emerge or fade away under different lights. These features can be distinct segments or other local quantities, such as gradient length or local variance, etc. This allows control of visibility of several features focusing and defocusing the attention of the user.

Since we are creating and assigning materials (and lights) with certain properties, we can incorporate our ideas within any transfer function design that has been proposed so far. Instead of assigning RGB values, one can assign material spectra to certain data values. Hence we can integrate these ideas into the framework proposed by e.g., either Kniss [13] or Pekar [26].

3.3 Navigating Visibility

We can design materials and lights such that each light influences exactly one material. Hence a linear combination of the lights will create a mixed rendition of the scene. Assuming that we have n materials and hence n lights, we have a n-dimensional space to navigate to produce different visualizations. This is potentially a difficult, if not impossible, task.

One way to proceed is to have a separate intensity-slider for each light attached. This is an important interaction method and is hence implemented in our program (see Fig. 3a). While changing the light source can also change the overall illumination of the scene, one might want to normalize by the overall brightness. This can then be controlled by a separate intensity slider. This interaction method provides versatile control because any combination of lights can be represented. Hence one is guaranteed to be able to explore the entire n-dimensional visualization space.

Since for some applications this completeness is not relevant, it is worthwhile to consider a simpler, but possibly more restrictive interaction method. For some problems it is not necessary to set a specific combination of lights. Instead it is enough to smoothly interpolate between certain key constellations. That especially applies to the interactive exploration of a volumetric rendering. The pure lights are most interesting and any combination in between can be chosen somewhat arbitrarily. Another aspect is that controlling all sliders requires several clicks and careful aiming for the slide button.

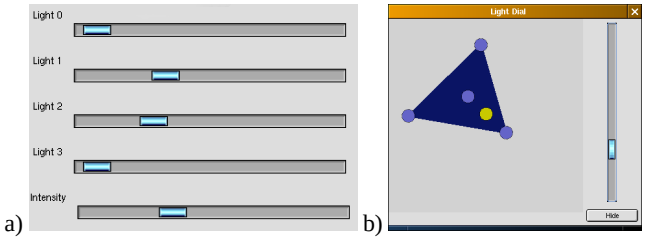


Figure 3: Different interfaces to control the mixture of lights a) using separate sliders for each light b) using normalized inverse distances of the slider (yellow circle) to the light nodes (blue circles).

A more intuitive interaction metaphor would include all light sources within an interaction widget. For this purpose we suggest a two-dimensional n-gon slider, mapping a 2D space into the n-dimensional parameter space (see Fig. 3b). Our n light sources are the vertices of the n-gon. Any position the plane characterizes a weighted sum of the light sources that make up the n-gon. This weighted light is weighted by the inverse distances to the light sources marked by the vertices. When the position coincides with the position of a light source in the widget, then the influence of all other light sources is eliminated. Hence this widget allows one to focus on one particular material attached to this light source at a time. The subspace of lights achieved this way can be further controlled by moving the control points around or switching them off, thus making the subspace take on a new 'shape'. This makes the slider more versatile. In particular, Fig. 1c shows the change from Fig. 1b after switching off the light that illuminates the flesh of the frog.

Our new proposed interaction widget implements three different interaction metaphors:

1. move light slider (change weights by distance),

2. move light nodes (change shape of 2D mapping in higher-dimensional parameter space, correlate dimensions by spatial distance in slider plane),

3. switch lights on/off (reducing dimensionality, comparable to taking a light node and moving it away; for deactivation the light can keep its position in the mapping plane.)

This type of 2D slider may be useful for other tasks, such as navigating between different points of view, controlling alpha values of different segments, etc. However, this is left for future work.

4 Real-Time Exploration

In the previous section we introduced a new way of changing the appearance of a scene. By smoothly fading between different light colours, segments become visible or merge with the background. To make this technique usable for data exploration it is important to re-render the scene at interactive framerates. The basic idea of *post-illumination* is to separate shading and compositing from the actual computation of an RGB colour. This allows one to generate new images for different light sources without repeating the entire rendering process. After all light bouncing and transmission is complete, only one matrix multiplication per final image pixel is needed to calculate the updated RGB image for the scene. This gives the necessary speedup for our purpose.

The following derivation is based on a general form of the rendering integral, which is discussed in Max [20]. This form combines different absorption and scattering properties up to multiple scattering. As shown by Kajiya [10], special cases of such an integral can be interpreted to describe surface raycasting and radiosity. This makes it a convenient framework to show the usability of post-illumination in general. For the implementation the integral has to be simplified and discretized. As well, the spectra cannot be represented as arbitrary continuous functions. Instead a linear colour model is used as described in § 5.

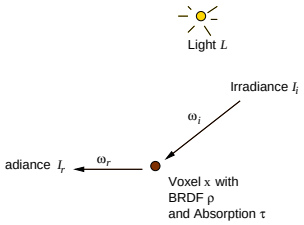


Figure 4: Lights and directions necessary to calculate the illumination at a point $\mathbf{x}$.

To incorporate a spectral colour model the rendering integral gets an additional dependency on the wavelength λ. Fig. 4 illustrates some symbols used in the following equations. The light I_i arriving at a point $\mathbf{x}$ from direction $\boldsymbol{\omega}_i$ is determined by

$$I_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) = \int_0^D I_r(\mathbf{x} - s\boldsymbol{\omega}_i, \boldsymbol{\omega}_i, \lambda) \cdot e^{-\int_0^s \tau(\mathbf{x}-t\boldsymbol{\omega}_i,\lambda)dt} ds, \quad (1)$$

where D is the distance from where the ray first enters the volume. The integral accumulates the light I_r radiated from a point $\mathbf{x}$ into direction $\boldsymbol{\omega}_i$. It is also known as the volume rendering integral, used in raycasting. An image pixel can be seen as a point $\mathbf{x}$ in the scene. When tracing a ray through the volume to the pixel the light is diminished by local absorption of the materials, which is represented by the extinguishing coefficient τ. This local absorption is independent of the wavelength if only 'flat' particle scattering is adopted. In case of a participating medium, the absorption varies depending on the wavelength of the light. So the light is not only diminished but can also change colour. If a simple local illumination model is

used, I_r is basically the light from the source weighted to incorporate diffuse reflection. To be more general and include shadowing and multiple scattering, the locally radiated light becomes

$$I_r(\mathbf{x}, \boldsymbol{\omega}_r, \lambda) = \int_{4\pi} \rho(\mathbf{x}, \boldsymbol{\omega}_r, \boldsymbol{\omega}_i, \lambda) \cdot \Big(I_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) + I_i^L(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) \Big)\, d\boldsymbol{\omega}_i. \quad (2)$$

The incident light is integrated over a unit sphere of all directions $\boldsymbol{\omega}_i$. The amount that is scattered towards $\boldsymbol{\omega}_r$ is described by a ratio ρ called the bidirectional reflection distribution function (BRDF). This function represents the reflection properties of the material. It determines the colour by reflecting wavelengths differently. Two different contributions to the incident light can be distinguished. The indirect light I_i results from multiple scattering from other particles. It is similar to the light illuminating a screen pixel. If the rendering method only traces primary scattering events this term is zero. In that case only direct light I^L contributes to the local scattering. It can be factored into the pure light spectrum L from the source and an extinction integral $\widetilde{I_i^L}$ by

$$I_i^L(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) = L(\lambda) \cdot \widetilde{I_i^L}(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) \quad (3)$$

$$\widetilde{I_i^L}(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) = \begin{cases} e^{-\int_0^1 \tau(t\mathbf{x}+(1-t)\mathbf{x}^L, \lambda)dt} & \text{if } \boldsymbol{\omega}_i = \mathbf{x}^L - \mathbf{x} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The factor $\widetilde{I_i^L}$ can also be seen as the amount of light from a flat white light source at position $\mathbf{x}^L$ arriving at point $\mathbf{x}$. This takes care of tracing shadows in the volume. In case of a local illumination model, shadows are ignored and I_i^L is equal to L.

Solving the recursion of I_i through I_r is subject to radiosity algorithms [20]. However, it is easy to see that the factorization propagates through the volume. Hence, in order to extract the actual light spectrum from the rendering integral, the factorization from Eq. 4 can be used directly in Eq. 2,

$$I_r^L(\mathbf{x}, \boldsymbol{\omega}_r, \lambda) = L(\lambda) \cdot \widetilde{I_r^L}(\mathbf{x}, \boldsymbol{\omega}_r, \lambda) \quad (5)$$

$$\widetilde{I_r^L}(\mathbf{x}, \boldsymbol{\omega}_r, \lambda) = \int_{4\pi} \rho(\mathbf{x}, \boldsymbol{\omega}_r, \boldsymbol{\omega}_i, \lambda) \cdot \Big(\widetilde{I}_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) + \widetilde{I_i^L}(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) \Big)\, d\boldsymbol{\omega}_i, \quad (6)$$

and subsequently in Eq. 1:

$$I(\mathbf{x}, \boldsymbol{\omega}, \lambda) = L(\lambda) \cdot \int_0^D \widetilde{I_r^L}(\mathbf{x} - s\boldsymbol{\omega}, \boldsymbol{\omega}, \lambda) \cdot e^{-\int_0^s \tau(\mathbf{x}-t\boldsymbol{\omega})dt} ds. \quad (7)$$

This is the basic equation for post-illumination. To be able to factor out the light spectrum there must not be any self emitting particles in the volume. Otherwise it would not be possible to factor out L. All light that is contributing from within, originates from the outside light source. Also, the illumination model is limited to work based on a BRDF which is just a spectral factor. Shifting or remapping the spectrum as must be done for fluorescence is not possible. A nice feature is that this method can still contain factors for diffuse and specular shading or anisotropic reflection. The BRDF can also be used to incorporate iridescent reflection models such as interference colours. In that case the re-weighting of different wavelengths is due to pathlength in a thin surface. It can be treated like a reflectance that depends on location $\mathbf{x}$ and the angles $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_r$.

Our implementation uses raycasting, only looking at primary scattering events. The actual computation of the rendering integral in Eq. 7 is done stepwise for samples $\mathbf{x}_i$ in a simplified form, where the $\mathbf{x}_i$ are defined by the discretization Δs, i.e., $\mathbf{x}_i = \mathbf{x}(s_i) = \mathbf{x} - s_i\boldsymbol{\omega} = \mathbf{x} - i\Delta s\boldsymbol{\omega}$. The colour spectrum C^L arriving in a pixel $\mathbf{x}_0$ for a light source L is accumulated by

$$C^L(\lambda) \approx L(\lambda) \left(\sum_{i=0}^{d} T(\mathbf{x}_i, \lambda) \cdot R^L(\mathbf{x}_i, \lambda) \cdot S^L(\mathbf{x}_i, \lambda) \cdot \Delta s \right). \quad (8)$$

Note that the BRDF has become a factor R^L. It still depends on the local position. The superscript symbolizes that it is a reflectance dedicated to the light source L. That means this factor is not the actual BRDF but an evaluated version for a certain angle of incidence from the light source and a radiation direction to the viewer. The contribution of the light from the outside is combined in S^L, which is the sum of the indirect light I_i and the shadowed direct irradiation $\widetilde{I_i^L}$. For a local illumination model this factor is just constant 1, since the light does not undergo any changes before illuminating the point $\mathbf{x}_i$.

The transmittance $T(\mathbf{x}_i, \lambda)$ is related to the discretized extinction integral. It is stepwise multiplied beginning with $T(\mathbf{x}_0, \lambda) \equiv 1$,

$$T(\mathbf{x}_{i+1}, \lambda) = T(\mathbf{x}_i, \lambda) \cdot A(\mathbf{x}_i, \lambda) \quad \text{with} \quad A(\mathbf{x}_i, \lambda) = e^{-\|\Delta\mathbf{x}_i\| \tau(\mathbf{x}_i, \lambda)}. \quad (9)$$

The local absorption $A(\mathbf{x}_i, \lambda)$ has to be adjusted to the sampling distance Δs. It can be precomputed for every voxel if the samples are equidistant. The following equation is an example of contributions to the light source dependent reflectance R^L:

$$R^L(\mathbf{x}_i, \lambda) = R(\mathbf{x}_i, \lambda) + R^L_{iridescent}(\mathbf{x}_i, \lambda) + w^L_{specular} \quad (10)$$

The reflectance R is the basic reflectance without any relationship to a light source. It is responsible for the actual colour of the material. The summand $w^L_{specular}$ is the specular coefficient from the Phong illumination model. It can be interpreted as adding a certain amount of flat white to the local reflectance, which will become the colour of the light source after multiplying L. Since everything is being calculated for a static light position and fixed perspective it is even possible to interpret interference colours as a local 'reflectance'. This and other iridescent effects such as dispersion or anisotropy can be combined in $R^L_{iridescent}$.

4.1 Post-Illumination with a linear colour model

To make the implementation efficient the discretized rendering integral has to be rewritten based on a linear colour model. In a linear subspace, illumination calculations are typically performed by a matrix multiplication. Peercy shows how to set up a matrix $\mathbf{M}$ that does the calculation within a linear subspace [25]. This matrix is devised for a special set of reflectances and is used for the multiplication of the weighted basis functions for the light and the reflectance. The result is projected back into the linear space. Here, the matrix $\mathbf{M}(y)$ is used to do this spectral multiplication in the linear space for a given reflectance y. That is, $x' = \mathbf{M}(y)x$ is the spectrum x after bouncing off reflectance y. This matrix is used to rewrite Eq. 8:

$$c^l = \left(\sum_{0}^{d} \mathbf{T}(\mathbf{x}_i) \cdot \mathbf{M}(r^l(\mathbf{x}_i)) \cdot \mathbf{M}(s^l(\mathbf{x}_i)) \right) \cdot l \quad \text{with} \quad \mathbf{T}(\mathbf{x}_{i+1}) = \mathbf{T}(\mathbf{x}_i) \cdot \mathbf{M}(a(\mathbf{x}_i)), \quad \mathbf{T}(\mathbf{x}_0) = \mathbf{I} \quad (11)$$

The symbols $\mathbf{T}$, a, r^l, s^l, and l are the the linear subspace representations for the full spectra of T, A, R^L, S^L, and L from Eq. 8.

The stepwise absorption of the light is treated as discrete transmission events with spectrum a. These are accumulated into a transparency matrix $\mathbf{T}$. Finally, for each pixel a matrix is added up along the ray through that pixel from local light transform matrices. A light can then be multiplied by that matrix afterwards to get the actual colour spectrum.

In general, we wish to reduce matrix multiplications to scalar multiplication. We use a special basis to promote simple componentwise multiplies of basis coefficients (the *factor model*) as described in § 5. The basis is designed to reduce $\mathbf{M}$ to a diagonal matrix, for which it is enough to just store and modify a diagonal vector. The step from complete matrix multiplications to simple componentwise multiplications of light vectors greatly improves the performance of the algorithm. Without any further optimizations the rendering speed is comparable to that of a RGB raycaster. The important difference is that the use of spectra yields correct colours and design flexibility, and post-illumination allows fast relighting.

5 Implementation Issues

5.1 The factor model of colour

The linear colour model we use here is based on an extension to spectral bases of the idea of *spectral sharpening* in colour constancy algorithms [7]. In graphics, usually RGB is simply multiplied to form colour. In surface graphics, for example, we multiply illuminant RGB times surface RGB, to generate a product RGB. This is not physically accurate, but can be made more so by a preliminary matrix transformation of camera sensors to generate an intermediary colour space.

Here, we are interested in multiplying spectra. The best we can do in an optimal fashion for representing spectra that participate in image formation is to form some principal component basis for the spectral curves. However, since each spectrum is represented as a sum over basis coefficients, this implies a matrix multiply of current light coefficients times the next interaction spectrum coefficients. But we can pre-"sharpen" the basis by a simple matrix transform and agree to operate within the sharpened basis for all surface or volume interactions [5]. 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 indeed we can represent spectral interactions in terms of the low-dimensional coefficients (typically 5 to 7D) and calculate interactions using simple componentwise multiplication of the coefficients. Since light colour change is now simply one componentwise multiplication, it can be the final multiply, and hence a post-illumination. In the last step, descending to 3D RGB colour, we simply need a $3 \times k$ matrix multiply for each pixel to create colour on the screen, where k is the dimension of our "colour" space.

5.2 Spectral Design

Section 3.1 provides guidelines for creating lights and materials for spectral transfer functions. We have developed a framework that creates spectra that fit given design properties [2]. The input to the algorithm is a design palette as shown in Fig. 5. It is a matrix of colours having columns for the light sources and rows associated with the materials. The colour for certain light-material combinations can be specified at the corresponding position in the matrix. To relax the constraints for the design process an additional weight indicates the importance of the specified colour. Each spectrum can either be input to the algorithm or be open for redesign. The example in Fig. 5 shows the setup for the frog in Fig. 1. The first light is standard illuminant D65, the second is Tungsten. The other two lights, as well as the materials are generated by the algorithm to match the specified colours. The first blue column indicates that all

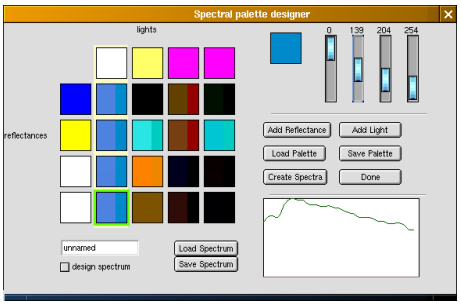


Figure 5: Interface to the spectral design palette

materials are metameric under light 1. The black fields indicate that some materials disappear under the corresponding light. The light in the last column for instance turns all materials to black except the second one, which is used to mark the eyes of the frog.

The algorithm allows us to find actual spectra for the given RGB colours. To generate 'reasonable' spectra it is possible to incorporate additional desirable properties. Smoothness is used to avoid spiky zig-zag spectra and also usually removes large and small spectral components, obviating the need for explicit upper and lower bounds. For rendering, the spectra are projected into the linear basis vector subspace described in §5.1. To keep the approximation error small, we include a matrix that minimizes the difference of the RGB values for the full and the projected spectrum. All properties (colours, smoothness, and error minimization) are expressed as a collection of normal equations. For these, a global minimum solution can be found using the pseudoinverse of a stacked matrix. This solution can then be used as the initial value for a constrained minimization that ensures that all components are positive.

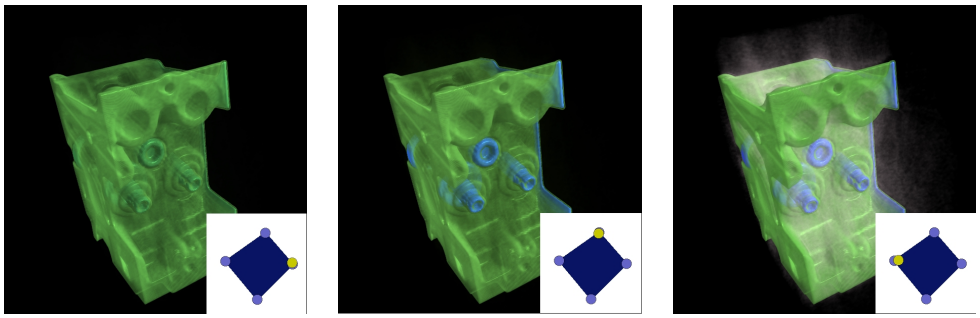
6 Results

To use re-illumination as a new degree of interaction imposes distinct restrictions on visualization. But the limitations inherent in this technique — in the form of defined lights for defined materials — are not necessarily negative. If arranged in a sensible way, information usually becomes more accessible to the user. By preparing different visual situations an author can *guide* the interaction that is provided through the light slider. This is a means to emphasize a certain set of features that can be explored by the user.

In Fig. 6 (a) the engine block and inner parts are metameric, with the 'smoke' (probably reconstruction noise) being black and thus invisible. Picture (b) shows the palette constellation for the next light, yielding distinguishable parts. In picture (c) the smoke becomes visible, glowing white. Using the light dial, it is possible to move smoothly between the lights. With the block keeping a constant colour this allows toggling the visibility of additional details.

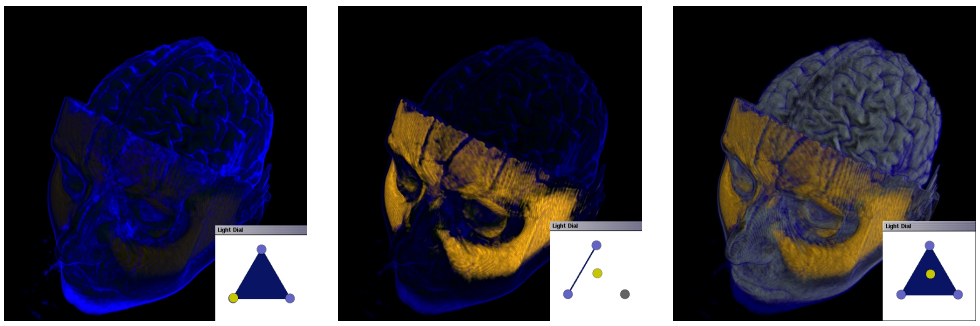
For the dataset in Fig. 7 each material has one dedicated light source. If one light is illuminating the scene the other segments go black. This allows emphasizing different aspects separately. In Fig. 7a the lights illuminating the inner segments have no influence since the slider is directly over one light node. Fig. 7b and 7c are both the result of a mixture of lights, with the light for the interior switched off in (b).

Rendering the frog (250x235x138) on a 1.4 GHz Pentium 4 takes 18s for a 800x800 image. The re-illumination is done at 7 fps, which depends on the image size only. A similar configuration takes 14s using an RGB colour model, which does not allow re-illumination. Rendering to a 400x400 screen takes 5s for the spec-



(a) (b) (c)

Figure 6: An engine, demonstrating the use of metamers and metameric blacks in spectral volume rendering.



(a) (b) (c)

Figure 7: Under each light only one material is not black. The mixture of lights produces different combinations of segments without repeating the raycasting.

tral and 4s for the RGB colour model. The post-illumination can be done at 20 fps. This shows that the RGB raycasting is an expected factor faster than spectral rendering. But the important difference is that the latter provides correct colours and light changing at framerates an order of magnitude higher.

7 Future Work

Our current implementation represents a proof of concept; the new technique has not been fully explored as of yet. We would like to incorporate this framework into some more sophisticated transfer function designs, e.g., Kniss' work [13]. Our ideas are not restricted to any rendering method. Hence we would like to include faster rendering as well. Some preliminary work has been done on splatting. Using vertex shaders, we are able to adapt hardware accelerated splatting to our spectral models.

The efficient representation of spectra using the *factor model* allows versatile modifications to the spectra while rendering. To fully exploit this fact it would be worthwhile investigating the integration of the method presented into light mapping mechanisms [8] or BRDF factorizations [21] to allow sophisticated spectral reflection models in real-time. This could open a new field for rendering photo-realistic volumetric scenes, which could be of great benefit for medical training applications.

Spectral materials could be used to more sensibly map information from multi-modal data to the screen. The material editor, in connection with the new exploration method of re-illumination, is a first step in that direction.

Another area to explore is the extension to spectral light fields using the technique introduced in [17]. Some preliminary work on re-lighting light fields is described in [22]. A problem is that the datasets are strongly related to the static lighting under which they have been acquired. But re-lighting is precisely what the method presented here is best used for. Work in this field has some interesting relationships to representing BRDFs as carried out by [8]. There has not been much work done so far on interactively rendering spectral BRDFs. A lot of work has been done on deriving elaborate spectral reflectance models, but no real-time capable (or convincingly general) representations have been developed as yet.

Another interesting area is user-oriented design. The interface widgets created here have not been evaluated with respect to their usefulness and we have played with several other ideas to interact with the n-dimensional light space. More work should be done to evaluate our interface widgets, develop new ones, and explore other possible metaphors for user interaction. Spectral volume rendering drastically complicates the transfer function design space. It is already difficult to explore transfer functions using RGB-based colour spaces.

8 Conclusions

In this particular work, and in volume rendering in general, the focus is not photo-realistic rendering. We have shown how to use concepts, like metamers and colour constancy, in order to explore a volumetric data set. These effects are not possible by using the traditional RGB colour model.

Based on an improved linear model, a framework for spectral volume rendering has been developed. To emphasize the use of re-illumination we proposed a method for rendering spectral scenes independent from a light source. This allows the user to get different impressions of the data under different lights, which is an enhancement of any technique, that utilizes a transferfunction for volume exploration. The spectral design method allows the preparation of specific illumination situations to highlight certain aspects

of the data. The artificial creation of spectra is complementary to using real spectra.

The design framework helps to make spectra more usable in computer graphics in general. For many applications it is essential to have control over specific properties of spectra while leaving others 'realistic'. There is an interesting duality inherent in spectral rendering: it provides improved realism on the one hand, and on the other provides enhanced possibilities for artificially manipulating a scene. These properties make spectra a powerful tool for computer graphics.

We have shown that our techniques are effective enhancements for transfer functions. While some of the demonstrated effects could perhaps be achieved by changing of opacities and assigned colours in the a traditional RGB-based rendering system, a complete re-rendering of the scene becomes necessary. Post-illumination proves useful for the interactive exploration of visualizations, even for high-quality rendering methods such as ray-tracing.

9 ACKNOWLEDGMENTS

This work was made possible by the partial support of Natural Sciences and Engineering Research Council of Canada (NSERC) and the Advanced Systems Institute (ASI) of BC. The authors would like to thank Ziemek Trzesicki for developing large parts of vuVolume – the visualization development environment we have used. Further we would like to thank the members of the Graphics, Usability, and Visualization (Gruvi) Lab for their creative input to this work.

REFERENCES

[1] C.L. Bajaj, V. Pascucci, and D.R. Schikore. The contour spectrum. In *IEEE Visualization 1997*, pages 167–175, November 1997.

[2] S. Bergner, T. Möller, and M.S. Drew. Spectral volume rendering. Technical Report SFU-CMPT-03/02-TR2002-03, Simon Fraser University, March 2002.

[3] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *1994 Symposium on Volume Visualization*, pages 91–98, October 1994.

[4] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 65–74, August 1988.

[5] M.S. Drew and G.D. Finlayson. Multispectral processing without spectra. Technical Report SFU-CMPT-03/02-TR2002-02, Simon Fraser University School of Computing Science, March 2002. Also: Representation of colour in a colour display system, UK Patent Application No. 0206916.9. Under review, British Patent Office.

[6] D.S. Ebert, C.J. Morris, P. Rheingans, and T.S. Yoo. Designing effective transfer functions for volume rendering from photographic volumes. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):183–197, June 2002.

[7] G.D. Finlayson, M.S. Drew, and B.V. Funt. Spectral sharpening: sensor transformations for improved color constancy. *J. Opt. Soc. Am. A*, 11(5):1553–1563, May 1994.

[8] W. Heidrich, H. Lensch, M. Cohen, and H.-P. Seidel. Light field techniques for reflections and refractions. In *Proc. of the EG Rendering Workshop '99*, 1999.

[9] G.M. Johnson and M.D. Fairchild. Full-spectral color calculations in realistic image synthesis. *IEEE Computer Graphics and Applications*, 19(4):47–53, July/August 1999.

[10] J. Kajiya. The rendering equation. In *Proc. of SIGGRAPH '86*, pages 143–150, 1986.

[11] S. Kilthau and T. Möller. Splatting optimizations. Technical Report SFU-CMPT-04/01-TR2001-02, School Of Computing Science, Simon Fraser University, April 2001.

[12] G. Kindlmann and J.W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *1998 Volume Visualization Symposium*, pages 79–86, October 1998.

[13] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *IEEE Visualization 2001*, pages 255–262, October 2001.

[14] G. Knittel. The ultravis system. In *2000 Symposium on Volume Visualization*, pages 287–294, October 2000.

[15] P. Lacroute and M. Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. In A. Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 451–458, July 1994.

[16] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, 8(3):29–37, May 1988.

[17] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. ACM SIGGRAPH 1996*. ACM SIGGRAPH, 1996.

[18] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[19] J. Marks, B. Andalman, P.A. Beardsley, W. Freeman, S. Gibson, J.K. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of SIGGRAPH 97*, pages 389–400, August 1997.

[20] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[21] M.D. McCool, J. Ang, and A. Ahmad. Homomorphic factorization of brdfs for high-performance rendering. In *Proceedings of ACM SIGGRAPH 2001*, pages 171–178, August 2001.

[22] D. Meneveaux and A. Fournier. Reshading light fields. In *SkiGraph 1999*, March 1999.

[23] H.J. Noordmans, H.T.M. van der Voort, and A.W.M. Smeulders. Spectral volume rendering. In *IEEE Transactions on Visualization and Computer Graphics*, pages 196–207, 2000.

[24] J. Orchard and T. Möller. Accelerated splatting using a 3d adjacency data structure. In *Graphics Interface 2001*, pages 191–200, June 2001.

[25] M.S. Peercy. Linear color representations for full spectral rendering. In *Computer Graphics (SIGGRAPH '93)*, pages 191–198, 1993.

[26] V. Pekar, R. Wiemker, and D. Hempel. Fast detection of meaningful isosurfaces for volume data visualization. In *IEEE Visualization 2001*, pages 223–230, October 2001.

[27] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The volumepro real-time ray-casting system. In *Proceedings of SIGGRAPH 99*, pages 251–260, August 1999.

[28] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 109–118, August 2000.

[29] G. Rougeron and B. Péroche. An adaptive representation of spectral data for reflectance computations. In *Rendering Techniques'97, (Proceedings of the Eurographics Workshop on Rendering)*, pages 126–138, 1997.

[30] T. Totsuka and M. Levoy. Frequency domain volume rendering. In *Proceedings of SIGGRAPH 93*, pages 271–278, August 1993.