A Practical Approach to Spectral Volume Rendering

Steven Bergner, *Student Member*, *IEEE*, Torsten Möller, *Member*, *IEEE*, Melanie Tory, and Mark S. Drew, *Member*, *IEEE Computer Society*

Abstract—To make a spectral representation of color practicable for volume rendering, a new low-dimensional subspace method is used to act as the carrier of spectral information. With that model, spectral light material interaction can be integrated into existing volume rendering methods at almost no penalty. In addition, slow rendering methods can profit from the new technique of postillumination—generating spectral images in real-time for arbitrary light spectra under a fixed viewpoint. Thus, the capability of spectral rendering to create distinct impressions of a scene under different lighting conditions is established as a method of real-time interaction. Although we use an achromatic opacity in our rendering, we show how spectral rendering permits different data set features to be emphasized or hidden as long as they have not been entirely obscured. The use of postillumination is an order of magnitude faster than changing the transfer function and repeating the projection step. To put the user in control of the spectral visualization, we devise a new widget, a "light-dial," for interactively changing the illumination and include a usability study of this new light space exploration tool. Applied to spectral transfer functions, different lights bring out or hide specific qualities of the data. In conjunction with postillumination, this provides a new means for preparing data for visualization and forms a new degree of freedom for guided exploration of volumetric data sets.

Index Terms—Volume rendering, real-time spectral computer graphics.

1 Introduction

Direct volume rendering allows views beneath surfaces, showing features inside volumetric data sets. To make certain parts of the scene visible, a transfer function assigns distinct colors and opacities (alpha) to the data points. Our approach is to use spectral reflectances instead of RGB values to set up the transfer function. Spectral materials in the scene now appear as different colors, depending on the illuminating light (although alpha compositing is not affected). This changes the appearance of the scene every time a new light is set. Under this framework, materials and lights can be specially designed to achieve certain visual effects. Since we use achromatic absorption, an object which is obscured by large regions of high opacity will not be made visible by relighting. It has been our experience, however, that the extra dimensions of information enabled by spectral rendering decrease the importance of opacity assignment and occlusion as the fundamental means of determining feature visibility. Changing the illumination of a scene thus becomes a new means for the user to interact with the visualization. In order to deploy this strategy as a useful tool for exploration, a new user interface is proposed.

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 data set simply by controlling the light sources. We show how to do this in real-time for any volume rendering algorithm. Extending the transfer function mapping to nonrestrictive spectra enables us to exploit to our advantage certain phenomena of color that are usually undesired. Our main idea is to make use of the concept of *metamers*. Differing light spectra that appear perceptually identical under a certain light are termed metameric. This means that such spectra differ only by a vector in the null space of the functions transforming spectra into colors (e.g., the spectral curves of the R, G, and B sensors of a virtual camera). By assigning metameric colors to certain data intensity ranges (using a *spectral transfer function*), we have the ability to visually merge these ranges into one color under a specific light. By changing the light spectra, we can visually separate these ranges. This has the potential to help the user explore the relationships between certain intensity intervals or materials (see Section 4.1 for more details).

In the following, we distinguish between light, reflectances, and colors. Color is a psychophysical quantity of light that is sensed by the human eye. As depicted in Fig. 1, light is sent out by sources in the environment and eventually reflected or refracted by other objects until it arrives at the eye. For graphics, we can use the *spectral power distribution* (SPD) of the electro-magnetic (EM) wave—the wavelength-dependent magnitude of the EM field's Poynting vector [20]—with visible wavelength range 400 nm to 700 nm, instead of full optical physics. When talking about

- *S. Bergner, T. Möller, and M.S. Drew are with the School of Computing Science, Simon Fraser University, Burnaby BC, V5A 1S6 Canada. E-mail: {sbergner, torsten, mark}@cs.sfu.ca.*
- *M. Tory is with the Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver BC, V6T 1Z4 Canada. E-mail: melanie@cs.ubc.ca.*

Manuscript received 27 Jan. 2004; revised 7 Sept. 2004; accepted 13 Sept. 2004; published online 13 Jan. 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0012-0104.

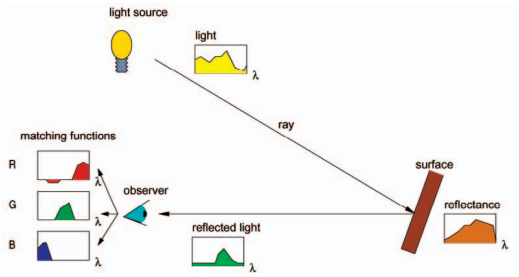

Fig. 1. Light may reflect off many surfaces before it reaches the eye, where it is perceived as "color."

physical correctness and error, we have to keep in mind that, for visualization, expressiveness of images comes before photorealism or physical accuracy. Furthermore, there are typically data sets, such as CT or MRI, that do not contain sufficient information on the original object. However, spectra can be used to great benefit in visualization. In particular, their additional dimensionality allows for more versatile image manipulation. This is the central idea behind the interaction we propose.

Spectral color models aspire to preserve the necessary physical properties of spectra to allow illumination calculations without *perceivable* errors. To minimize computational costs, we use a new low-dimensional linear representation called the *spectral factor model* of color, described in Section 3.

Our new technique of postillumination, described in Section 5, allows one to efficiently rerender a scene for changing light sources. For the interactive manipulation of a spectral rendition via our postillumination scheme, a new "light-dial" widget is devised in Section 4.2. This paper is an extension of our previous work [2]. Here, we 1) show details on the spectral factor model (Section 3), 2) extend various rendering algorithms to incorporate postillumination (Section 6), and 3) present a usability study, including a keystroke-level analysis as well as a usability inspection of the new interface (see Section 7).

2 Related Work

2.1 Volume Rendering

The past five years have seen great advances in the area of real-time volume graphics. Much-improved consumer hardware (referred to as GPU) is available that has been efficiently exploited for volume rendering using texture mapping capabilities as well as many new features of modern graphics cards [6]. Other improved data structures, such as an adjacency list and an RLE-based data structure, were introduced to speed up the splatting algorithm tremendously [11], [17]. Fourier Volume Rendering [14], [21] enables software-only real-time rendering (with quality degradation). Our method of image-based spectral relighting can be incorporated into existing volume rendering methods, as discussed later in Section 6. A performance comparison shows that the speed of projection for arbitrary viewpoint is only slightly compromised gaining interactive light manipulation.

To change the color scale in the rendered image, Kaneda et al. [10] linearly combine renditions of different Fourier basis transfer functions. This requires multiple rendering passes. Their approach is restricted to recoloring pseudocolor images, while we seek to combine correct spectral light-material interaction with fast rerendering.

Finally, Noordmans et al. [16] have integrated a spectral color model into the volume rendering pipeline. To speed up their rendering, they go from rendering full/actual spectra to rendering coefficients of materials that are assumed to occupy distinct spectral bands. The difference in our approach is to gain performance by using an optimized linear color 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 in Noordmans et al. Our method also provides more freedom in the way spectra change while interacting within the scene. This is essential for more advanced illumination models. Further, we discuss guidelines, not addressed by Noordmans et al., of how to sensibly create and use artificial spectra, an important issue for volume rendering and for computer graphics in general.

2.2 Representing Light

The most straightforward method of dealing with spectral information is to simply carry spectrally point-sampled data through any rendering calculation. Sufficiently sampling at a uniform interval, however, would mean too many samples.[1]

Linear models express an SPD as a weighted sum of basis functions. They 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 [18] 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.

3 Spectral Factor Model

We present a new color-vector representation that adopts the strength of linear models while avoiding spectral multiplication: Light interaction is reduced to a simple extension of RGB-color componentwise multiplication—a *spectral factor model* [5].

The linear color model we use here is based on an extension of the idea of using *spectral sharpening* in color constancy algorithms in computer vision [8] to spectral bases. In graphics, we usually multiply RGB componentwise to form color. In surface graphics, for example, illuminant RGB is multiplied times surface RGB in order to generate a product RGB. When lighting changes, we would like to model RGB change as a simple diagonal transform. This is not physically accurate, but can be made more so by a preliminary matrix transformation of camera sensors to generate an intermediary color space. This transform is, in fact, included in many digital cameras and is called "spectral sharpening" since camera sensors are

1. Real surfaces and illuminants are typically measured at more than 100 sample points and represented using 31 samples from 400 nm to 700 nm at 10 nm intervals.

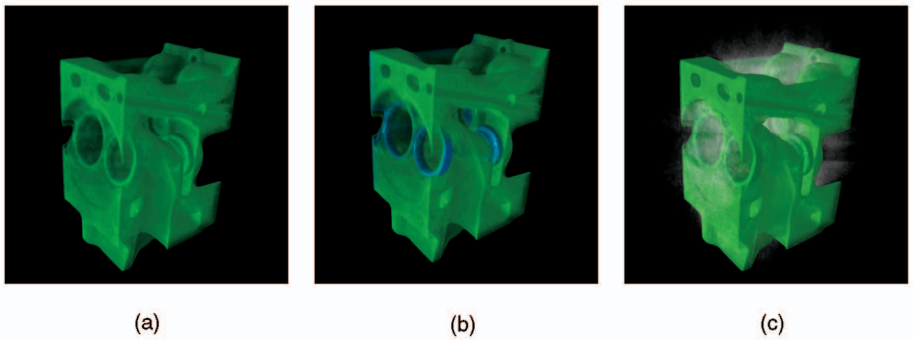

Fig. 2. **Spectral raycasting** of the engine data set under three different lights. The parts of the engine have three different materials assigned. The images were generated using postillumination—no rerendering took place, just an image-based postprocessing.

thereby made effectively more narrowband. For multiple light-surface interactions, however, such a simple transform will not lead to enough accuracy and we must fall back on full spectra.

Here, we can apply the same idea to the linear model basis set: We form combinations of the basis set vectors that are optimally narrowband. 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 [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 five to seven) and calculate interactions using componentwise multiplication of the coefficients. Using the new basis, the multiplication of spectral functions in the linear subspace is reduced to a simple componentwise multiply involving the coefficients ε_i of the current light, $E(\lambda)$, times coefficients σ_i for the next interaction surface, reflectance, or transmission function $S(\lambda)$. If

$$C(\lambda) \;=\; E(\lambda)S(\lambda), \tag{1}$$

then, projected into the d-dimensional linear subspace, we have approximately

$$\gamma_i \;\simeq\; \varepsilon_i\,\sigma_i, \quad i = 1..d, \tag{2}$$

where γ_i are the basis coefficients for the color signal $C(\lambda)$. Note that instead we could simply be using sums of delta function spectra similarly, but then would be giving up the power of the basis set to accurately model actual spectra. Here, we retain simplicity and low dimensionality without abandoning physical spectra.

4 Interaction Icon

While material assignment has been used successfully 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 chosen or *designed* with close attention to their use in volume rendering: Both metamers, with $C_1(\lambda)$ and $C_2(\lambda)$ yielding the same RGB, 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 distinct perceptual discrimination.
3. Finally, we are able to navigate through different visualizations by controlling the light source. Since multiple dimensions (n weights for n light spectra) are involved, we create a new interaction icon.

While the second step is a straightforward material-based transfer function design, Steps 1 and 3 deserve further discussion.

4.1 Material Design

By assigning distinct reflectances, we are able to separate the appearance of materials. However, we also have *metameric materials* that, under a particular light source, result in colors that are identical for our visual sensors. Hence, we have the ability to merge certain materials and, therefore, guide the user's attention toward or away from these materials. Fig. 2 illustrates this effect. Under light 1 (Fig. 2a), both materials of the block look the same. The entire engine is perceived as a homogeneous structure. Blending to light 2 (Fig. 2b), the metamer effect breaks down and the materials become distinguishable. This process of newly emerging details is likely to be noticed by the user.

Another principle in spectral material design is to make use of the effect of *color constancy*. In Fig. 2, we have created light sources, in connection with materials, that will leave the color of one material the same and change only the color of the materials we intend to influence with that light—the engine block keeps its green color under both lights.

Since traditional color theory usually describes only phenomena that occur in the real world, this restricts us to nonartificial metameric materials. However, it would be desirable to design materials that disappear entirely from the image, if so desired by the user. In color science, these materials are called *metameric blacks* [23], i.e., materials with zero RGB under some specific set of lights. Such a material is assigned to the surrounding reconstruction noise in Fig. 2. This "smoke" becomes visible as we blend over to light 3 in

Fig. 2c. In the other images, it is rendered as well, but, under the first two lights, it remains black. In rendering, alpha compositing still takes place, so occlusion of other materials is, in fact, not 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. This idea is applied in our implementation of spectral Fourier volume rendering discussed in Section 6.

While metamers can be used to reduce complexity in images, constant colors are useful when blending from one light source to another. Having some things change their appearance while others remain the same is likely to grab the observer's attention. Thus, such materials can be used to guide the user's attention to certain aspects of the data during interactive exploration. Within the same visualization (without any change of the transfer function), we are able to emphasize and deemphasize particular regions of interest in connection with particular light sources. Therefore, we can help the user understand the relationships between these materials.

Having an idea of what color constellations are desirable for visualization, we now have the task of finding spectra for lights and materials that fit these criteria. One approach is to use materials previously acquired from real-world scenes. Nevertheless, for most applications, it would be impractical to have to look for real materials and lights that fit the desired properties (metamerism, color constancy, metameric blacks). For that purpose, we employ a design procedure [1] that combines real spectra with new artificial but physically-based ones, generating a set of lights and reflectances.

4.2 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 m materials and n lights, we have an n-dimensional space to navigate to produce different visualizations. This is a difficult task for the user.

We developed an interaction metaphor that includes all light sources within one interaction widget, which we call the "light dial." This is a two-dimensional n-gon slider, mapping a 2D space into the n-dimensional parameter space (see Fig. 3). Our n light sources are the vertices of the n-gon. Any position on the plane characterizes a weighted sum of the light sources that make up the n-gon. This way, the higher-dimensional parameter space is mapped to a 2D topology of nodes.

Each control node has a position on the screen and represents a light source and the mouse is used to freely move the mixture selector over and between the nodes (e.g., the yellow dot position in Fig. 3). Position is used to determine a scalar weight for each node, with a value that grows as the selector comes closer to a node. We can compute a weight $L_i(\mathbf{x})$ applied to the light for a selector position x in the following manner:

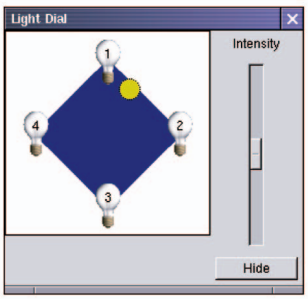

Fig. 3. **The light dial**—interface to control the mixture of lights using normalized inverse distances of the mixture selector (yellow circle) to the light nodes (bulb icons) (see (3)).

$$L_i(\mathbf{x}) = \prod_{k=1,\, k\neq i}^{n} \frac{\|\mathbf{x} - \mathbf{x}_k\|}{\|\mathbf{x}_i - \mathbf{x}_k\|}. \tag{3}$$

To make the weights usable for combining light sources, we normalize the sum of all weights L_i to one and scale it with a separate intensity slider. When the position of the dot coincides with the position of a light bulb in the widget, the influence of all other light sources is eliminated. The influence of a source can be entirely removed by switching it off. Hence, the light dial allows one to focus on the influence of a single light to the materials used in the scene. It has not been designed to navigate the full n-D space at once. Instead, the intention is to conveniently slide from one pure light to the next pure light.[2] Additionally, it is possible to correlate dimensions (making lights have similar intensity) by moving two light bulbs closer to each other. This modifies the shape of the 2D mapping of the parameter space and allows the user to reach new parameter constellations.

To summarize, our light dial implements three different interaction metaphors:

1. move yellow dot (change weights by distance)—the object is weighted toward the lights we want to see it under,
2. move light nodes—group lights, or drag lights away to lower their influence (mathematically speaking: change shape of 2D mapping of higher-dimensional parameter space, correlate dimensions by decreasing spatial distance in widget plane), and
3. switch lights on/off (reducing dimensionality, comparable to taking a light node and moving it far away; deactivation of the light allows it to keep its position in the mapping plane.)

This makes the 2D dial a convenient interface to navigate through the color schemes. The light dial metaphor may be useful for other tasks, such as navigating between different points of view, controlling alpha values of different

2. Nevertheless, it is still possible to navigate the entire space of weight combinations. Imagine a concentric arrangement of the light bulbs around the light dot in the middle: All lights have the same weights. Now, a light bulb may be dragged instead of the dot, moved closer to or farther away from the center (where the dot is). Since only one distance is changing, it is possible to change only one weight. To actually have just one weight changing, the separate overall intensity slider would have to be moved correspondingly.

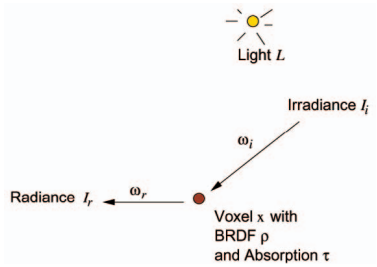

Fig. 4. Illustration of symbols and abbreviations used in Section 5.

segments, etc. A more thorough evaluation of this widget is provided in Section 7.

5 Postillumination

By fading between different lights, the whole scene changes its appearance. To make this technique usable for data exploration, we propose the method of *postillumination*. It allows us to rerender the scene at interactive frame-rates by separating shading and compositing from the actual computation of an RGB color. Fig. 4 illustrates the notation that is adopted in this section. Looking at the rendering integral as discussed by Max [15], it is possible to factor out the light source, yielding the following equation:

$$I_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) = L(\lambda) \cdot \widetilde{I}_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda). \tag{4}$$

It describes the irradiance I_i depending on the wavelength λ and the position $\mathbf{x}$ on a ray of length D from direction $\boldsymbol{\omega}_i$. In this model, the point $\mathbf{x}$ can either be a voxel in the volume or a pixel on the projection screen at which we collect the incoming light. The second term, $\widetilde{I}_i$, is a ratio indicating which portion of light L is transferred through the scene. It is calculated as

$$\begin{aligned}\widetilde{I}_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) = \widetilde{I}_i(\mathbf{d}, \boldsymbol{\omega}_i, \lambda) T(\mathbf{x}, \boldsymbol{\omega}_i, \lambda, D) + \\ \int_0^D \widetilde{I}_r(\mathbf{x} - s\boldsymbol{\omega}_i, \boldsymbol{\omega}_i, \lambda) \cdot T(\mathbf{x}, \boldsymbol{\omega}_i, \lambda, s) ds,\end{aligned} \tag{5}$$

where $T(\mathbf{x}, \boldsymbol{\omega}, \lambda, s) = e^{-\int_0^s \tau(\mathbf{x} - t\boldsymbol{\omega}, \lambda) dt}$ is the extinction integral that accounts for the effect of absorption τ. The first term is the irradiance from outside the volume at a point $\mathbf{d} = \mathbf{x} - D\boldsymbol{\omega}_i$. Since the light source $L(\lambda)$ is factored out from the integral, the radiance I_r is modified to become a ratio $\widetilde{I}_r$. This ratio can be understood as the amount of irradiance that is scattered from direction $\boldsymbol{\omega}_r$. More precisely, it is formed as

$$\begin{aligned}\widetilde{I}_r(\mathbf{x}, \boldsymbol{\omega}_r, \lambda) = \\ \int_{4\pi} \rho(\mathbf{x}, \boldsymbol{\omega}_r, \boldsymbol{\omega}_i, \lambda) \cdot \\ \left(\widetilde{I}_i(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) + \widetilde{I}_i^L(\mathbf{x}, \boldsymbol{\omega}_i, \lambda) \right) d\boldsymbol{\omega}_i,\end{aligned} \tag{6}$$

containing the product of the bidirectional reflection distribution function ρ (BRDF) and the irradiance integrated over all unit directions $\boldsymbol{\omega}_i$. Here, the shadowed amount of direct light from source L is captured by $\widetilde{I}_i^L$ and the indirectly scattered light is the recursive term $\widetilde{I}_i$ from (5).

Our implementation uses raycasting with a local illumination model. This simplifies $\widetilde{I}_r^L$ to just become the reflectance of the local material, weighted by the Phong shading coefficients (ambient, diffuse, and specular). Even anisotropic and iridescent effects could be incorporated here as a wavelength dependent weighting. As for the local illumination, $\widetilde{I}_i$ is zero for all directions other than along the ray $\boldsymbol{\omega}_i = \boldsymbol{\omega}_r$. Furthermore, our shadowing ratio $\widetilde{I}_i^L$ is constant 1 for all voxels and directions (no shadowing). An example rendition incorporating the use of specular lighting is shown in Fig. 2, where we can observe specular highlights at the solid inner rings inside the engine. The images are generated using an achromatic absorption (alpha blending). Nevertheless, the use of metamers and metameric blacks enables us to influence the visibility of additional structures, even after the spectral image has been generated, by simply changing the illuminating light spectrum.

The postponed multiplication of the light source allows one to generate new images for different light spectra without repeating the projection and compositing. After all light bouncing and transmission is complete, only one matrix multiplication in the linear color model per final image pixel is left to calculate the updated RGB image. This gives the necessary speedup for interactive reillumination.

6 Spectra in Volume Rendering Techniques

Many benefits accrue to different rendering methods from incorporating a spectral color model. Here, we explain how several classic techniques (a nonexhaustive list) are impacted by spectral extensions.

6.1 Raycasting

This technique is a straightforward implementation of a simplified version of the rendering integral discussed in Section 5. Raycasting provides high rendering quality, but, due to random memory access, it is not very fast in its standard implementation. This technique derives the biggest profit from using postillumination. Rendering reflectances to the screen takes only slightly longer than RGB rendering [2]. Once the spectral image is generated, it can be illuminated with different light sources via a 3×7 matrix transformation per pixel, combining the illumination matrix and the color transform to produce RGB. The examples previously shown in Fig. 2 illustrate the visual effect of postillumination applied to the raycast data set of an engine.

6.2 Splatting

Splatting is typically a very fast volume rendering method, introduced by Westover [22]. Current graphics cards are based on RGB compositing. This is the major constraint when using spectral rendering. In order to combine fast splatting with spectral light material interaction, we have to limit the rendering to flat absorption (alpha blending). There are two ways to build a spectral splatter. One is to perform a multipass rendering until all channels of the color

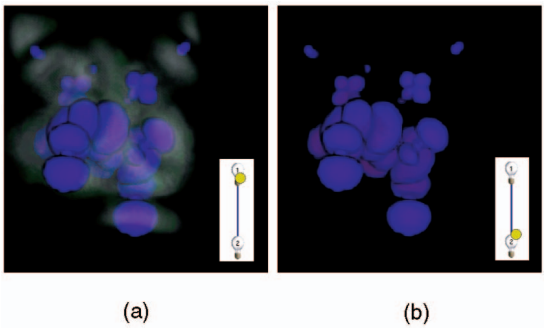

Fig. 5. **Spectral Splatting** of hipiph data set. The lower intensities are assigned a low-alpha material (a) glowing white under light 1 or (b) black under light 2.

model have been projected to the screen (e.g., two passes for a six channel factor model). The resulting image could then be postilluminated just as discussed above. An alternative approach is to simply use an RGB transfer function whose colors are made from spectral reflectances and the active light source. If the color (RGB) for a voxel is determined, it can be splatted to the screen using standard graphics hardware.

One way to further improve performance is to use vertex programs to do the illumination calculation at each voxel with spectral light sources and perform the color transformation to RGB. The color of the light source, its position, and the color transformation matrix (size $3 \times (7+1)$) are stored in the vertex program parameter space. The 7-component voxel reflectance is passed, utilizing the primary and secondary color registers. The conversion from the 7-component model can be done by splitting the conversion matrix horizontally into two 4×4 matrices (last column is zero). These are applied to the halves of the reflected light color vector accordingly. By adding the results the final color is obtained. Such operations are basic in vertex programs and can be done in a few lines of code. Using the spectral splatter, the $64 \times 64 \times 64$ hipiph data set in Fig. 5 can be rendered at 18 frames per second, still allowing for an interactive manipulation of the transfer function and spectral light color. A similar RGB splatter without spectra is only slightly faster, as can be seen in Table 1.

6.3 Fourier Volume Rendering

The basic idea of Fourier volume rendering (FVR), as introduced by Malzbender [14], is to calculate the projection of a data set to the screen by extracting a slice of the volume in the frequency domain. This makes FVR a very powerful technique, especially for large data sets. Because the intensities are simply added up, the result is an x-ray-like image. Totsuka and Levoy [21] further improved the technique by incorporating hemispherical shading and depth-cueing. Recent extensions by Entezari et al. even allow for specular highlights [7].

To integrate spectral illumination into FVR, our approach is to render the seven components of the linear color model separately. Postillumination can be carried out by projecting only the reflectances and performing the illumination calculation (multiplying the light spectrum) afterward in the image. Examples are shown in Fig. 6 and Fig. 7. The first figure makes extensive use of metameric blacks to toggle visibility of different segments. Spectral materials are assigned such that metamers make different segments merge together and segments colored with metameric blacks entirely disappear from the image sum for a certain light source. FVR is a fast technique for pure software implementation. Using spectra and postillumination in FVR allows reinterpreting the spectra in the transfer function in a fast image-based operation and makes an expensive rerun of the preprocessing unnecessary. Specific parts of the data can be separated, merged, or faded out without changing the transfer function. This not only improves the visual quality of the renderings, but also adds a new degree of interaction to FVR.

6.4 Summary

An evaluation of the performance of different spectral rendering techniques is provided in Table 1. The techniques are, from left to right: spectral raycasting and separate postillumination (PI) in comparison with plain RGB raycasting, spectral splatting versus RGB splatting, and spectral Fourier volume rendering (SFVR). All timings were taken on a 2.4 GHz Pentium 4 with a GeForce 4 using a constant image size of 512×512. The speed is given in frames per second for projecting a frame/reilluminating a frame. The last column also contains time for preprocessing for SFVR. PI is independent of the rendering technique. Nevertheless, the timings are given for images from the spectral raycasting. The method has been optimized to only consider pixels of nonzero reflectance. Thus, the screen *fill* correlates to the PI timings. Note that PI is an image-based operation and not necessarily faster than object based methods as is the case for the spectral splatting of *hipiph*. As discussed above, our variant of the spectral splatter recomputes the projection for reillumination because compositing is done in RGB. Note that the images of SFVR internally do not actually have the size of the display but,

TABLE 1
Performance of Different Spectral Rendering Techniques (in fps), Computing Images from a New Viewpoint

Dataset	Fill	Spec-RC	PI-RC	RGB-RC	Spec-Splat	RGB-Splat	Spec-FVR/PI-FVR
hipiph	97%	0.22	9.2	0.24	18.5	22.2	50/166 (1s)
tomato	72%	0.15	11.6	0.17	0.78	1.7	5.8/34.5 (85s)
uncbrain	62%	0.15	13.3	0.17	0.3	0.4	15.8/30.3 (85s)
engine	79%	0.13	10.7	0.15	0.5	0.6	8.7/47.6 (85s)
frog	69%	0.08	12.0	0.09	0.31	0.38	*(n/a)*

Fill *denotes the amount of nonblack screen pixels.* PI *is postillumination (for fixed viewpoint) on the spectral image. Data set dimensions: hipiph* 64^3, *tomato* $256^2 \times 64$, *uncbrain* $256^2 \times 145$, *engine*, $256^2 \times 128$, *and frog* $500 \times 470 \times 138$.

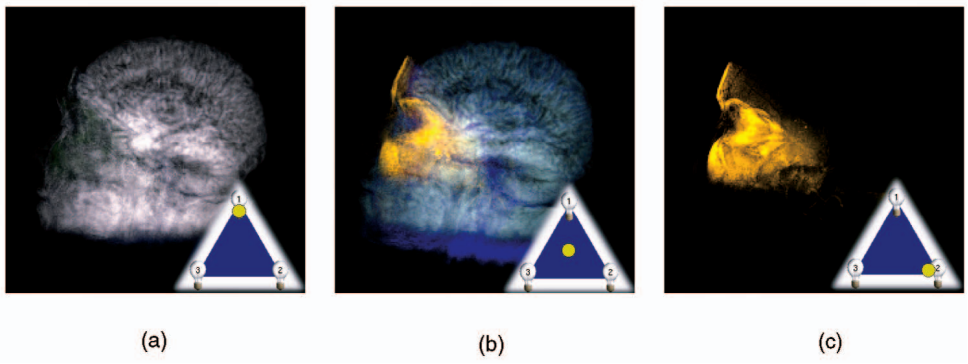

Fig. 6. **Spectral Fourier Volume Rendering** of the UNC-Brain data set (CT-scan). Each light illuminates one material, keeping the other two black. The images show three different mixtures of lights illuminating: (a) interior only, (b) three different regions, and (c) frontal skull. As to the x-ray character of the images, the black material entirely disappears from the image (no occlusion).

instead, are obtained from the extracted slice from the frequency cube. Thus, the timing of PI on SFVR is not directly comparable to the spectral raycasting. The two major conclusions that can be drawn from the table are: 1) The performance of spectral rendering compared with corresponding RGB techniques (e.g., spectral splatting) shows very competitive framerates and 2) the use of postillumination is an order of magnitude faster than changing the transfer function and repeating the projection step.

7 Usability Assessment of the Light Dial

To utilize spectral volume rendering, users would typically 1) set up materials and lights prior to a visualization session and then 2) examine data by setting up transfer functions and manipulating lights to change visibility (as established in Section 4). Designing a good interface for setting up materials and lights is complex; our interface has the appropriate functionality but nonideal usability. We leave development of a good material/light designer for future work. Designing good transfer function editors is also challenging, but is not unique to spectral volume rendering. Hence, we focus on evaluating our proposed light-dial for navigating visibility.

We evaluated the light dial and its conventional counterpart—a multiple-slider interface having a separate intensity slider for each light. The following two methods of evaluation were chosen:

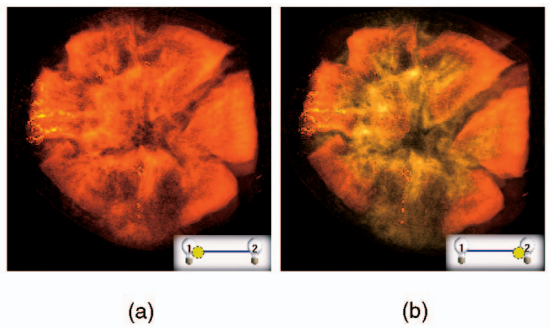

Fig. 7. **Spectral Fourier Volume Rendering** of a tomato (MRI) rendered with (a) two metameric materials, and (b) the same materials being nonmetameric under a different light.

1. keystroke-level analysis [3] to compare interface speed and
2. expert assessment of the interfaces via usability inspection [13].

7.1 Keystroke-Level Analysis

In keystroke-level analysis, tasks are broken into keystroke-level actions (clicking a key, dragging the mouse, etc.). The time for an expert user to complete a task is predicted by adding together average times for the keystroke-level actions [3].

We made several assumptions in our keystroke-level analysis:

1. Some actions (e.g., homing hands on the mouse) are the same for both interfaces and can therefore be ignored.
2. Both interfaces are constrained such that the largest mouse movement is 5 cm.
3. An average mouse movement is half the largest movement.

We used component actions and times from [3] to predict the time required to adjust settings in each interface and considered several situations (e.g., turning one or all light(s) on). We considered the interfaces with and without shortcuts (i.e., clicking at the desired end location without dragging). Shortcuts work for any slider in the multiple-sliders interface and the yellow slider in the light dial, but not for light sliders in the light dial since it would be impossible to know which light a user intended to move.

Results favor the light dial when only a few lights are active. With four lights or less, the light dial is equal to or faster than multiple-sliders for all tasks. With more than four lights, the light dial is as good as or better than multiple-sliders when only the yellow slider must be moved (i.e., to turn on one light, all lights, or sets of lights that are close together and far from undesired lights). These results are the same whether shortcuts are allowed or not.

Setting up constellations with the light dial is inefficient because dragging lights takes time. However, as previously mentioned, navigating between constellations with the light dial is efficient. Since materials and lights are designed to match, pure lights (one light source alone) may be most interesting and arbitrary light combinations may be used

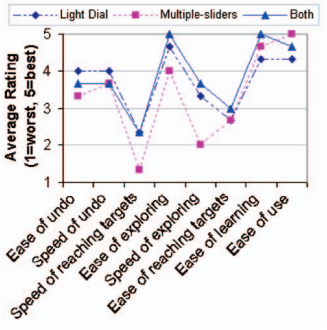

Fig. 8. Average ratings in heuristic evaluation.

less often. Furthermore, users who do set up new light dial "shapes" may then spend substantial time moving the main slider within that shape before altering the shape again. In this case, the cost of setting up the shape may be offset by the speed gained during the second navigation phase. However, users who set up arbitrary light combinations quite often are faster with the multiple-slider interface.

7.2 Usability Study

7.2.1 Method

Three usability experts were asked to assess the interfaces together and separately using usability inspection techniques [13]. All three experts were graduate students in human-computer interaction at Simon Fraser University. None had prior knowledge of the spectral volume rendering project. Evaluations were conducted independently.

Lights and materials were set up ahead of time. A 4-light setup was used. Experts were given descriptions of typical end users and typical tasks (data exploration, matching the display to target images, and returning to previous settings). They answered questions about advantages, disadvantages, and problems with the interfaces and evaluated them with respect to the following heuristics: ease of learning, of use, of exploring a data set, of matching target images, ease of undo, speed of exploring, speed of reaching target images, and speed of undo. Heuristics were assessed on a 5-point rating scale and other questions were open-ended. After a written report was completed, evaluators discussed their opinions with the experimenter.

7.2.2 Usability Inspection Results

Fig. 8 displays average ratings in the heuristic evaluation. Multiple sliders were easier to learn and use (overall), probably because of their familiarity. Exploration and undo were easier with the light dial, probably because a wide range of light levels could be explored by simply moving one slider and relative positions of controls were easier to remember with the 2D representation. In terms of speed, the light dial was rated faster for all tasks, agreeing with the keystroke-level analysis. Having both interaction techniques available side-by-side generally got good ratings and appears to be a good compromise. In this case, a user can choose which interface to use and the other updates automatically.

All evaluators agreed that both interfaces should be available since each is useful for different tasks. The multiple-sliders interface was considered useful for understanding the contribution of each light and for resetting these contributions exactly during an undo operation. The light dial was useful for quickly exploring, turning lights on and off, and navigating preset constellations. However, judging the levels of each light in the light dial was difficult. Some evaluators favored having both interfaces in the same dialog box, but others suggested that the light dial be the main interface and the multiple-slider interface be accessible as an auxiliary widget. One report predicted that prolonged use of the light dial would cause less wrist strain than the multiple-slider interface. Overall, the usability study indicated that users can perform basic lighting changes via the light dial with little training.

8 Discussion and Future Work

To construct a spectral transfer function from a set of given materials is just as straightforward as it is to create a traditional RGB-based visualization. The important difference is that the manipulation of the light source allows for more versatile interactions later on. Nevertheless, creating one's own spectra still requires substantial work and a number of deliberate decisions. The actual benefits that justify the additional work can be summarized as follows:

1. The scene may bear more information than it is possible to see with three-dimensional RGB.
2. The interaction with the scene is more versatile since light changes become a means of exploration.
3. The user's attention may be guided by fading between different visualizations that are inherent to the incorporated spectra.
4. The visualization may more truthfully reflect the physical behavior of given materials.

After the projection step is done, different impressions may arise from the visualization by postilluminating the image with different lights. The user can interact with the image by manipulating the light. Thus, through one spectral scene setup, different messages may be conveyed to the observer. Developing this method up to a state of interactive frame rates for any data and renderer is seen as a major contribution of this paper.

A different method of achieving a similar effect is to render the data set with different traditional (RGB-based) transfer functions and to blend between the resulting images. Despite the fact that this method may require more rendering passes, it is a valid alternative to our approach. The important difference to postillumination is that we can apply any kind of light spectrum to the unilluminated image. This is not possible when combining prerendered images of RGB-based transfer functions. In our current framework, this advantage is not really used. The only situation where this can actually be observed is when designing new lights and seeing their immediate effect on the scene without rerendering. This is possible within the framework, but has not been further evaluated as of yet.

The system presented here opens opportunities for different types of applications. One direction would be photorealistic volume rendering. As our method computes correct colors for light-material interactions, it is possible to incorporate real-world materials into the

scene. This could be useful to create more convincing training scenarios for medical applications. For this, it might be interesting to incorporate the novel ideas of spectral subsurface scattering [12].

To date, not much work has been done on interactively rendering spectral BRDFs. Elaborate spectral reflectance models have been derived [9], [19], but they are all lacking real-time capacity or perhaps even general representation of spectral reflectance properties. Besides increasing realism, it is possible to scale the amount of information contained in an image. Additional details may gradually be revealed as different lights illuminate the scene. The use of spectra increases the dimensionality of the output image. The actual look of the visualization may be determined by choosing an appropriate light. To make this choice more manageable, our proposed "light-dial" widget restricts the illuminant to be a combination of predefined light spectra. The light dial also offers a very simple interaction and exploration metaphor once the materials and lights are set up by an expert. So, the method could be very good for educational applications (medical and otherwise), museum exhibits, etc.

8.1 Conclusions

In volume rendering in general, the focus usually is not photorealistic rendering and that has not been the case in this paper either. Nevertheless, the method is based on physical models rooted in spectral measurements. We have shown how to use concepts such as metamers and color constancy in order to explore a volumetric data set. These effects are not possible by using traditional RGB.

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

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 colors in the traditional RGB-based rendering system, a complete rerendering of the scene becomes necessary. Postillumination proves useful for the interactive exploration of visualizations, especially for high-quality rendering methods such as ray-tracing.

Acknowledgments

The *hipiph* data set is courtesy of Scripps Clinic, California. The *uncbrain* is courtesy of the University of North Carolina Chapel Hill. The *tomato* is from Berkeley Laboratory. The *frog* is part of VTK by Kitware (TM). The *engine* is available at www.volvis.org

References

[1] S. Bergner, T. Möller, and M. Drew, "A Spectral Engine for Visualization," Technical Report SFU-CMPT-TR2004-06, School of Computing Science, Simon Fraser Univ., June 2004.

[2] S. Bergner, T. Möller, M.S. Drew, and G.D. Finlayson, "Interactive Spectral Volume Rendering," *Proc. IEEE Visualization 2002,* pp. 101-108, Oct. 2002.

[3] S.K. Card and T.P. Moran, "User Technology: From Pointing to Pondering," *Readings in Human Computer Interaction: Toward the Year 2000,* R.M. Baecker, J. Grudin, W.A.S. Buxton, and S. Greenberg, eds. pp. 587-602, San Francisco: Morgan Kaufmann, 1995.

[4] R.A. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics (Proc. SIGGRAPH '88),* vol. 22, pp. 65-74, Aug. 1988.

[5] M.S. Drew and G.D. Finlayson, "Multispectral Processing without Spectra," *J. Opt. Soc. Am. A,* vol. 20, pp. 1181-1193, 2003, also "Representation of Colour in a Colour Display System," UK Patent Application No. 0206916.9, under review, British Patent Office.

[6] K. Engel, M. Kraus, and T. Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading," *Proc. 2001 SIGGRAPH/Eurographics Workshop Graphics Hardware,* pp. 9-16, Aug. 2001.

[7] A. Entezari, R. Scoggins, T. Möller, and R. Machiraju, "Shading for Fourier Volume Rendering," *Proc. IEEE Volume Visualization and Graphics Symp. 2002,* pp. 131-138, Oct. 2002.

[8] G.D. Finlayson, M.S. Drew, and B.V. Funt, "Spectral Sharpening: Sensor Transformations for Improved Color Constancy," *J. Optical Soc. Am. A,* vol. 11, no. 5, pp. 1553-1563, May 1994.

[9] J.S. Gondek, G.W. Meyer, and J.G. Newman, "Wavelength Dependent Reflectance Functions," *Proc. SIGGRAPH 1994,* pp. 213-220, 1994.

[10] K. Kaneda, Y. Dobashi, K. Yamamoto, and H. Yamashita, "Fast Volume Rendering with Adjustable Color Maps," *Proc. 1996 Symp. Volume Visualization,* pp. 7-14, 1996.

[11] S. Kilthau and T. Möller, "Splatting Optimizations," Technical Report SFU-CMPT-04/01-TR2001-02, School of Computing Science, Simon Fraser Univ., Apr. 2001.

[12] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson, "A Model for Volume Lighting and Modeling," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 2, pp. 150-162, Apr.-June 2003.

[13] R.L. Mack and J. Nielsen, "Usability Inspection Methods: Executive Summary," *Readings in Human Computer Interaction: Toward the Year 2000,* R.M. Baecker, J. Grudin, W.A.S. Buxton, and S. Greenberg, eds., pp. 170-181, San Francisco: Morgan Kaufmann, 1995.

[14] T. Malzbender, "Fourier Volume Rendering," *ACM Trans. Graphics,* vol. 12, no. 3, pp. 233-250, July 1993.

[15] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 99-108, June 1995.

[16] H.J. Noordmans, H.T.M. van der Voort, and A.W.M. Smeulders, "Spectral Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* pp. 196-207, 2000.

[17] J. Orchard and T. Möller, "Accelerated splatting Using a 3D Adjacency Data Structure," *Proc. Graphics Interface 2001,* pp. 191-200, June 2001.

[18] M.S. Peercy, "Linear Color Representations for Full Spectral Rendering," *Computer Graphics (SIGGRAPH '93),* pp. 191-198, 1993.

[19] J. Stam, "Diffraction Shaders," *Proc. SIGGRAPH '99, Computer Graphics Proc., Ann. Conf. Series,* pp. 101-110, Aug. 1999.

[20] Y. Sun, F.D. Fracchia, M.S. Drew, and T.W. Calvert, "A Spectrally-Based Framework for Realistic Image Synthesis," *The Visual Computer: Int'l J. Computer Graphics,* vol. 17, no. 7, pp. 429-444, 2001.

[21] T. Totsuka and M. Levoy, "Frequency Domain Volume Rendering," *Proc. SIGGRAPH 93, Computer Graphics Proc., Ann. Conf. Series,* pp. 271-278, Aug. 1993.

[22] L. Westover, "Footprint Evaluation for Volume Rendering," *Computer Graphics,* vol. 24, no. 4, pp. 367-376, Aug. 1990.

[23] G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulas,* second ed. New York: Wiley, 1982.

Steven Bergner is a PhD student at Simon Fraser University. He received the diploma in computational visualistics from the University of Magdeburg in 2003. His research interests lie in the areas of scientific visualization and image processing. He seeks to understand the interactions between visual representations and the human observer to utilize them in creating meaningful visualizations. He is a student member of the IEEE.

Torsten Möller received the PhD degree in computer and information science from Ohio State University in 1999 and the Vordiplom (BSc) in mathematical computer science from Humboldt University of Berlin, Germany. He is an assistant professor at the School of Computing Science at Simon Fraser University. His research interests include the fields of scientific visualization and computer graphics, especially the mathematical foundations of visualization and graphics. He is codirector of the Graphics, Usability and Visualization Lab and serves on the Board of Advisors for the Centre for Scientific Computing at Simon Fraser University. He has been appointed vice chair for publications of the IEEE Technical Committee of Visualization and Graphics. He is a member of the IEEE, IEEE Computer Society, ACM, Eurographics, and CIPS.

Melanie Tory received the PhD degree from Simon Fraser University in 2004 and the BSc degree from the University of British Columbia in 1999. She is a postdoctoral fellow in the Imager Lab at the University of British Columbia. Her research objective is to enhance the value of visualization tools by developing and evaluating effective display designs, interaction techniques, and user interfaces.

Mark S. Drew received his undergraduate education in engineering science at the University of Toronto, studied functional analysis in mathematics for his master's degree, and received the PhD degree in theoretical physics from the University of British Columbia. He is an associate professor in the School of Computing Science at Simon Fraser University (SFU) in Vancouver, Canada. Combining work on energy systems and computer applications, he held an Industrial Postdoctoral Fellowship and, subsequently, an Industrial Research Fellowship in industry until he joined SFU. His interests lie in the fields of multimedia, computer vision especially focusing on image processing, color, photorealistic computer graphics, and visualization. He has published more than 80 refereed papers in journals and conference proceedings. He is the holder of a United States patent in digital color processing and a United Kingdom patent application in color computer graphics. He is a member of the IEEE Computer Society.