Interactive Display of Isosurfaces with Global Illumination

Chris Wyman, *Member*, *IEEE*, Steven Parker, *Member*, *IEEE*, Peter Shirley, and Charles Hansen, *Senior Member*, *IEEE*

Abstract—In many applications, volumetric data sets are examined by displaying *isosurfaces*, surfaces where the data, or some function of the data, takes on a given value. Interactive applications typically use local lighting models to render such surfaces. This work introduces a method to precompute or lazily compute global illumination to improve interactive isosurface renderings. The precomputed illumination resides in a separate volume and includes direct light, shadows, and interreflections. Using this volume, interactive globally illuminated renderings of isosurfaces become feasible while still allowing dynamic manipulation of lighting, viewpoint and isovalue.

Index Terms—Path tracing, isosurface, visualization, rendering, global illumination, precomputed radiance transfer.

1 Introduction

Isosurfaces are widely used in computer graphics and scientific visualization, whether to help model complex objects [1] or to reveal structure of scalar-valued functions and medical imaging data [2]. Because both computational and acquired data sets tend to be large, only recently has interactive display and manipulation of isosurfaces become feasible for full-resolution data sets [3]. As in most interactive visualization systems, the rendering of isosurfaces is based on only local illumination models, such as Phong shading, perhaps coupled with simple shadows. For partially lit concave regions, these simple shading models fail to capture the subtle effects of interreflecting light. These regions are typically dominated by a local ambient term which provides no cues to environmental visibility or reflections from nearby surfaces. Fig. 1 shows the details brought out by our approach, both in shadowed regions and those with high interreflections.

While we usually think of "direct lighting" as coming from a small light source, it can also come from extended light sources. In the extreme case, the entire sphere of directions is a light source and "shadowing" occurs when not all of the background is visible at a point. This results in darkening for concavities, exploited as *accessibility shading* [4], *obscurance shading* [5], and *ambient occlusion* [6]. More recently, direct lighting from a uniform area source has been applied to illuminating isosurfaces extracted from volumes to good effect [7].

In this paper, we extend the class of lighting effects for isosurfaces to include full global illumination from arbitrary light sources. Previous approaches are special cases of our technique. Our method requires an expensive preprocess, but does not greatly affect interactive performance, and can even speed it up since shadows can be computed as part of the preprocess. Our method uses a conventional 3D texture to encode volume illumination data. For static illumination and a static volume, a scalar texture encoding irradiance (similar to Greger et al. [8]) stores the necessary global illumination for all isosurfaces. For dynamic lights and complex materials, each texel stores an approximation of the precomputed radiance, as per Sloan et al. [9]. In either case, rendering interpolates between neighboring texels to approximate the global illumination on the correct isosurface.

2 Background

Rendering images of isosurfaces can be accomplished by first extracting some surface representation from the underlying data followed by some method of shading the isosurface. Alternatively, visualizing isosurfaces with direct volume rendering requires an appropriate transfer function and a shading model.

In practice, many volume data sets are rectilinearly sampled on a 3D lattice, and can be represented as a function ρ defined at lattice points $\vec{x}_i$. Some interpolation method defines $\rho(\vec{x})$ for other points $\vec{x}$ not on the lattice. Other data sets are sampled on a tetrahedral lattice with an associated interpolation function. Analytical definitions are possible for $\rho(\vec{x})$, often arising out of mathematical applications. Such analytical representations can easily be sampled on rectilinear or tetrahedral lattices.

Given a sampled data set $\rho(\vec{x}_i)$, the marching cubes algorithm [10] extracts explicit polygons approximating an isosurface $I(\rho_{iso})$ with isovalue ρ_{iso}, where $I(\rho_{iso}) = \{\vec{x} \mid \rho(\vec{x}) = \rho_{iso}\}$. Various improvements make this technique faster and more robust, but these improvements still generate explicit polygonal representations of the surface. Ray tracing and volume rendering provide an alternate method of displaying isosurfaces, which need not construct and store

- *C. Wyman is with The University of Iowa, 14 MacLean Hall, Iowa City, IA, 52242-1419. E-mail: cwyman@cs.uiowa.edu.*
- *S. Parker is with the Scientific Computing and Imaging Institute, University of Utah, 50 S. Central Campus Dr., Room 3490, Salt Lake City, UT 84112. E-mail: sparker@cs.utah.edu.*
- *P. Shirley and C. Hansen are with the School of Computing, The University of Utah, 50 S. Central Campus Dr., Room 3190, Salt Lake City, UT 84112. E-mail: {shirley, hansen}@cs.utah.edu.*

Manuscript received 16 July 2004; revised 8 Nov. 2004; accepted 8 Dec. 2004; published online 10 Jan. 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0076-0704.

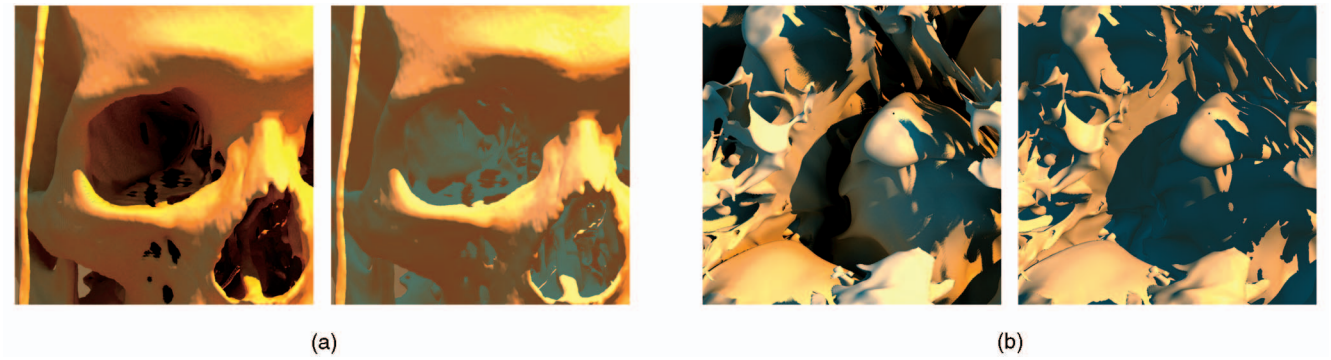

Fig. 1. The left image in each pair shows our approach, the right shows Phong with shadows. Note the improvement in regions dominated by indirect lighting, particularly in (a) the eye sockets and (b) the concavities in the simulation data.

a polygonal representation [3], [11]. Once an isosurface has been identified, standard illumination models can be applied to visualize the data.

Commonly, extracted isosurfaces are shaded using the Phong model [12] and variations using similar ambient, diffuse, and specular components. Such techniques give poor depth and proximity cues, as they rely on purely local information. Illumination techniques for direct volume rendering, such as those surveyed in Max [13], allow translucency and scattering along the viewing ray and shadow rays, but they fail to allow area lights and are not interactive. Sobierajski and Kaufman [14] apply global illumination to volume data sets, but they shade at runtime, so few effects can be incorporated while maintaining interactivity. Behrens and Ratering [15] use a slice compositing technique to interactively add shadows to volume renderings.

Several techniques have been proposed to interactively shade volumes with global lighting. The Irradiance Volume [8] samples the irradiance contribution from a scene, allowing objects moving around the scene to be shaded by static environment illumination. However, objects placed in the Irradiance Volume cannot interact with themselves, which is important for correct global illumination of isosurfaces. Precomputed radiance transfer techniques [9], [16], [17] can be used to shadow or cast caustics on nearby objects by sampling transfer functions in a volume around the occluder. Unfortunately, these techniques do not allow dynamic changes to object geometry, which is important when exploring the isosurfaces of volume data sets. Kniss et al. [18] describe an interactive volume illumination model that captures shadowing and forward scattering through translucent materials. However, this method does not allow for arbitrary interreflections and, thus, does not greatly improve isosurface visualization. The *vicinity shading* technique of Stewart [7] encodes direct illumination from a large uniform light in a 3D texture and allows its addition to standard local shading models while interactively changing the displayed surface. However, this method only provides an approach for direct illumination and does not incorporate indirect illumination.

Numerous approaches [19], [20], [21], [22] interactively apply global illumination in polygonal scenes. Generally, these approaches either perform lookups into illumination samples on two-dimensional surfaces or maintain interactivity by limiting the number and cost of paths traced each frame. Our technique takes a sampling approach with volumetric objects.

Concurrent, independent work by Beason [23] also examines precomputing illumination for isosurfaces of volumetric data. While Beason's work focuses on static illumination of isosurfaces, it examines a number of issues we do not, such as translucency, caustics, and other sampling strategies.

3 Overview

A brute-force approach to globally illuminating an isosurface would compute illumination at every point visible from the eye. Fig. 2 shows how a Monte Carlo path tracing technique would perform this computation. Obviously, performing such computations on a per-pixel basis quickly becomes cost prohibitive, so caching techniques [24] are usually preferable. Many existing techniques cache radiance [25], irradiance [8], or more complex radiance transfer functions [9], [16], [26] to speed illumination computations.

In volume visualization, users commonly change the displayed isovalue, thereby changing the isosurface, to view different structures in the volume. Most illumination caching techniques are object specific, so as the surface changes new illumination samples must be computed. We propose a technique which stores either irradiance or more complex transfer functions in a 3D texture coupled with the volume. Each texel t corresponds to some point $\vec{x}_t$ in the volume. Our

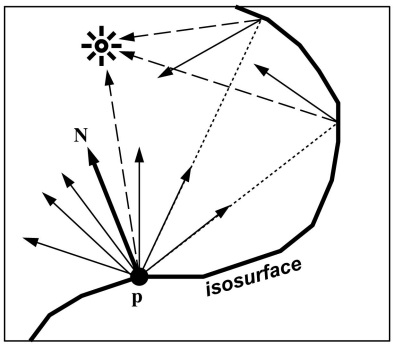

Fig. 2. Computing the irradiance at point $\vec{p}$ involves sending a shadow ray and multiple reflection rays. Reflection rays either hit the background or another part of the isosurface, in which case rays are recursively generated.

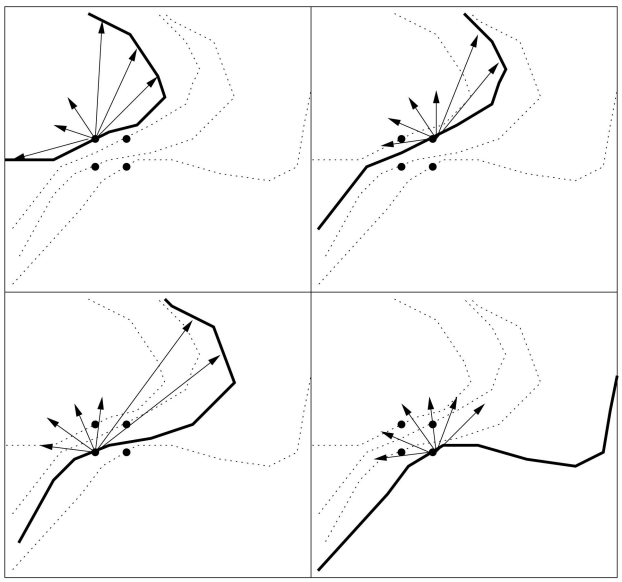

Fig. 3. Global illumination at each texel t is computed using standard techniques based on the isosurface $I(\rho(\vec{x}_t))$ through the sample.

process is broken into two steps. During the computation step, illumination values can be computed using any standard global illumination technique. For each texel t, we extract the isosurface $I(\rho(\vec{x}_t))$ running through $\vec{x}_t$. Using this isosurface, the global illumination is computed at point $\vec{x}_t$ via standard approaches and stored in texel t. Fig. 3 shows how this works for four adjacent samples using a Monte Carlo sampling scheme. During the rendering step, we interpolate between cached texels to the correct isosurface, allowing the interactive display of arbitrary isosurfaces.

As with most illumination caching techniques, the rationale is that global illumination generally changes slowly for varying spatial location. In our method this assumption applies in two ways: Illumination should change gradually over a surface and illumination should change gradually with changing isosurfaces.

One situation exists where this approximation obviously breaks down—hard shadows. Hard shadows involve a very visible discontinuity in the direct illumination. As shown in Fig. 3, each sample in our illumination lattice is potentially computed on a separate isosurface. In regions where a shadow discontinuity exists, some samples will be in shadow and others will be illuminated. Using an interpolation scheme to compute the illumination results in a partially shadowed point between samples. Thus, for a static isosurface, results near shadow edges will be blurred, similar to the effect achieved with percentage closer filtering [27].

Since we desire to dynamically change the rendered isosurface, we must also examine the effect of such changes on shadow boundaries. As the displayed isosurface changes, the interpolated illumination value on the surface changes linearly with distance from illumination samples, just as the illumination varies over a single isosurface. Thus, faint shadows may exist when there is no apparent occluder or small occluders may cast no shadow. The effect is that shadows will "fade" in and out as occluders appear and disappear. Examples of these effects can be seen in Fig. 4. The scene is illuminated by a blue and brown environment

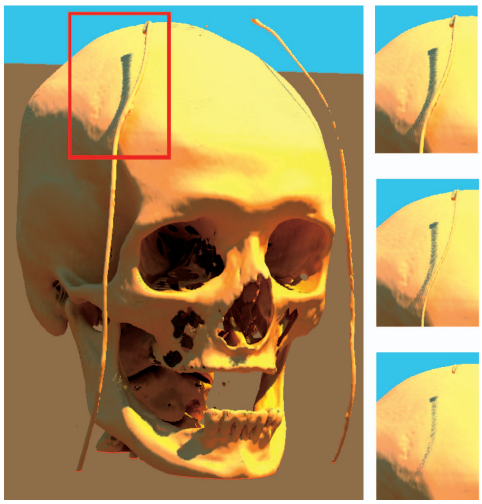

Fig. 4. The Visible Female's skull globally illuminated using our technique. The right images show how the cord's shadow fades out with increasing isovalues.

with a yellow point light source. The shadows are blurred slightly, and as the isosurface changes the cord fades out before its shadow.

Note that because sharp discontinuities in direct illumination are most visible, hard shadows are a worst-case scenario. For soft shadows or indirect illumination, the perceived effects are less pronounced, so our approximation of a smoothly changing illumination function becomes more accurate. While artifacts similar to those in Fig. 4 may still occur, they will be less noticeable.

The artifacts that occur with changing isovalues arise from our approximation of the nonlinear global illumination function using simple trilinear interpolation. This approximation is what causes the "fading" of shadows and the faint banding seen in our images.

While this assumption occasionally causes problems, our technique provides significantly more locality information than local models, especially in concavities and dark shadows where ambient terms either provide little or conflicting information. Additionally, since shadows are included in our representation, they require no additional per-frame computation. In fact, our approach renders faster than simple Phong and Lambertian models when including hard shadows.

4 Algorithm

Our technique has two stages: illumination computation and interactive rendering. A simplistic approach would perform all the illumination computations as a preprocess before rendering. As this can require significant time and not all illumination data may be required, it is also possible to lazily perform computations as needed, assuming the display of some uncomputed illumination is acceptable until computations are complete.

4.1 Illumination Computation

Each sample in our illumination lattice stores some representation of the global illumination at that point. This illumination is described by the rendering equation [28]:

$$L(\vec{x}_t, \vec{\omega}) = \int_\Omega f_r(\vec{x}_t, \vec{\omega}, \vec{\omega}') L(\vec{x}_t, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}_t) d\vec{\omega}', \quad (1)$$

where $\vec{x}_t$ is the location of the illumination texel t with normal $\vec{n}_t$, $\vec{\omega}$ is the exitant direction, $\vec{\omega}'$ is the incident direction varying over the hemisphere Ω, f_r is the BRDF, and L is the radiance incident at $\vec{x}_t$ from $\vec{\omega}'$.

4.1.1 Computing Irradiance

Depending on an application's required materials and illumination, this equation and the representation stored in the illumination texture can be varied to reduce computation time and storage space. For a simple diffuse surface with fixed lighting, a single irradiance value is sufficient at each lattice point. In this case, the rendering equation can be rewritten as:

$$L(\vec{x}_t) = f_r(\vec{x}_t) \int_\Omega L(\vec{x}_t, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}_t) d\vec{\omega}' = \frac{R(\vec{x}_t) E(\vec{x}_t)}{\pi}. \quad (2)$$

Here, the diffuse BRDF has no dependency on $\vec{\omega}'$. It can be removed from the integral and is then equivalent to the surface albedo $R(\vec{x}_t)$ divided by π. The remainder of the integral is the irradiance at point $\vec{x}_t$, $E(\vec{x}_t)$. Storing $E(\vec{x}_t)$ in our texture allows for easy illumination during rendering, as per (2).

To compute the irradiance at each sample point $\vec{x}_t$, we use Monte Carlo pathtracing. Using N random vectors, $\vec{v}_j$, sampled over the hemisphere Ω, the irradiance is approximated:

$$E(\vec{x}_t) = \frac{1}{N} \sum_{j=1}^{N} L(\vec{x}_t, \vec{v}_j). \quad (3)$$

Using this equation, we compute the irradiance at every point as follows:

for all $\vec{x}_t$ in illumination lattice **do**
  compute isovalue $\rho(\vec{x}_t)$
  compute isosurface normal $\vec{n}_t$
  sample hemisphere Ω defined by $\vec{n}_t$
  for all samples $\vec{v}_i \in \Omega$ **do**
    compute illumination at $\vec{x}_t$ in direction $\vec{v}_i$ using isosurface with isovalue $\rho(\vec{x}_t)$
  end for
  compute irradiance at $\vec{x}_t$ using (3).
end for

Explicitly extracting different isosurfaces for each sample is quite costly. To avoid this cost, we analytically intersect the trilinear surface using a raytracer [29]. Unfortunately, trilinear techniques generate noisy surfaces and normals, which can significantly impact the quality of the computed global illumination (see Fig. 5). Rather than directly computing normals from the analytical trilinear surface or using a simple finite difference gradient, we use a normal defined by the gradient smoothed over a $4 \times 4 \times 4$ voxel region with a tricubic B-spline kernel. By smoothing normals and slightly offsetting $\vec{x}_t$ in the normal direction during illumination computations, we reduce this microscopic self-shadowing noise. Note the global illumination artifacts seen in Fig. 5 using gradient normals occur on both microscopic and macroscopic scales. Noise occurs on the microscopic scale due to aliasing on the trilinear surface. Artifacts occur on the macroscopic scale when the volumetric data set and the illumination volume have different resolutions. Visible bands occur where voxels from the two volumes align, as can be seen in the bottom left image in Fig. 5.

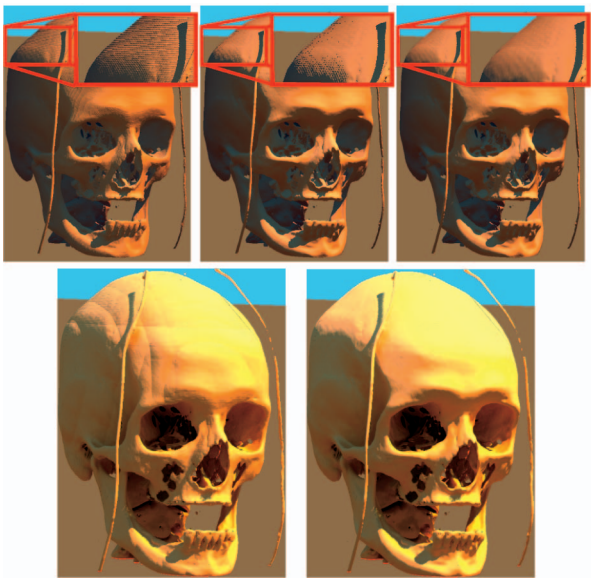

Fig. 5. An isosurface from the Visible Female's head extracted using analytical intersection of the trilinear surface. Top: Direct illumination from a point light using (left) gradient normals, (center) tricubic B-spline smoothed normals, and (right) offset surface with smoothed normals. Bottom: Four bounce global illumination using (left) gradient normals and (right) offset surface with smoothed normals.

For more complex effects such as dynamic illumination or nondiffuse material BRDFs, a simple irradiance value will not suffice, and a more complex representation of the illumination must be computed. We chose to use spherical harmonic (SH) basis functions to represent more complex illumination, since spherical harmonics efficiently represent low frequency lighting. But, the real advantages of using a SH basis are that integration of two functions simplifies to a dot product of their projected SH coefficients and projected functions can easily be rotated in the SH basis. This allows interactive rendering in scenes with dynamic illumination. Mathematical details of these properties are discussed further in the Appendix.

4.1.2 Computing Spherical Harmonic Coefficients

Assuming a diffuse material, incident illumination from a distant environment $L_\infty(\vec{\omega}')$ invariant over $\vec{x}$, and a visibility function $V(\vec{x}_t, \vec{\omega}')$, we can rewrite the rendering equation as:

$$L(\vec{x}_t) = f_r(\vec{x}_t) \int_\Omega \left[L_\infty(\vec{\omega}') V(\vec{x}_t, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}_t) + L(\vec{x}_t^{\vec{\omega}'})(1 - V(\vec{x}_t, \vec{\omega}')) \right] d\vec{\omega}'. \quad (4)$$

Note $\vec{x}_t^{\vec{\omega}'}$ is the point occluding $\vec{x}_t$ in direction $\vec{\omega}'$. As the incident illumination is assumed constant over the volume, it can be factored out of the recursive integrals when using the SH basis, leaving a geometry term $T(\vec{x}_t, \vec{\omega}')$, we call the *radiance transfer function*:

$$L(\vec{x}_t) = f_r(\vec{x}_t) \int_\Omega T(\vec{x}_t, \vec{\omega}\,')L_\infty(\vec{\omega}\,')d\vec{\omega}\,'. \quad (5)$$

Intuitively, the radiance transfer function T describes how light incident on the volume from direction $\vec{\omega}\,'$ affects the illumination at point $\vec{x}_t$. In our SH representation, we store the precomputed SH coefficients $\tau_i(\vec{x}_t)$ of T at each texel in our illumination texture and rotate the SH coefficients ℓ_i for L_∞ dynamically each frame. These coefficients are computed by projecting via:

$$\tau_i(\vec{x}_t) = \int_\Omega T(\vec{x}_t, \vec{\omega}\,')Y_i(\vec{\omega}\,')d\vec{\omega}\,' \quad (6)$$

$$\ell_i = \int_{sphere} L_\infty(\vec{\omega}\,')Y_i(\vec{\omega}\,')d\vec{\omega}\,'. \quad (7)$$

Using Monte Carlo integration to approximate these integrals can easily be performed with a pathtracer:

$$\int T(\vec{x}_t, \vec{\omega}\,')Y_i(\vec{\omega}\,')d\vec{\omega}\,' \approx \frac{1}{N}\sum_{i=1}^{N}\frac{T(\vec{x}_t, \vec{\omega}\,')Y_i(\vec{\omega}\,')}{p(\vec{\omega}\,')}, \quad (8)$$

where the functions T and Y_i are sampled N times, and $p(\vec{\omega}\,')$ is probability of randomly selecting direction $\vec{\omega}\,'$. To compute this radiance transfer function, we sample over the hemisphere Ω of visible directions at $\vec{x}_t$, so a uniform sampling gives $p(\vec{\omega}\,') = \frac{1}{2\pi}$. When sampling the incident illumination L_∞, the entire sphere is sampled, so for uniform sampling $p(\vec{\omega}\,') = \frac{1}{4\pi}$. Green [30] exhaustively explains this process of sampling T and L_∞, including pseudocode.

For nondiffuse materials the BRDF changes with viewpoint, adding an additional degree of freedom to the equation. Spherical harmonic transfer matrices can represent such material properties, as described in Sloan et al. [9], [17]. We do not present results for such materials, as more coefficients are necessary to capture the additional dimensionality introduced by viewpoint-dependent specular effects.

Alternate bases, such as wavelets, could similarly be used to represent the global illumination in our volume, especially if higher-frequency effects are desired. However, light reflected from diffuse materials consists of mainly low frequency information, so spherical harmonics efficiently represent the data.

4.2 Interactive Rendering

Once we have our illumination samples computed, rendering is straightforward. At every visible point on the displayed isosurface, we index into the illumination texture to find the eight nearest neighbors. We interpolate the coefficients stored in the texture, and use the interpolated coefficients for rendering.

When using an irradiance texture, we simply use (2) to compute the illumination based on the stored irradiance and the albedo. With the spherical harmonic representation, we interpolate the stored spherical harmonic coefficients τ_i and then perform a vector dot product with the environmental lighting SH coefficients ℓ_i. So, for nth-order coefficients:

$$L(\vec{x}_t) = f_r(\vec{x}_t)\sum_{i=0}^{n^2}\tau_i\ell_i. \quad (9)$$

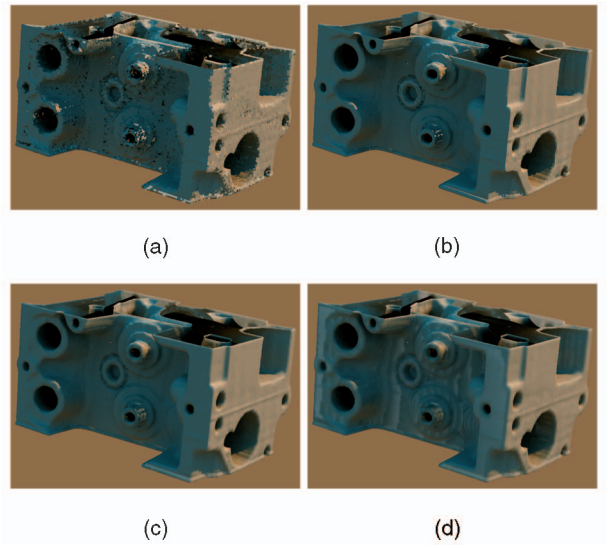

Fig. 6. Engine block illuminated using (a) the illumination sample with closest isovalue, (b) the nearest illumination sample, (c) a trilinearly interpolated value, or (d) a value computed with a tricubic B-Spline kernel.

We expected a higher order interpolation [31] scheme over nearby neighbors would be required to generate smooth illumination over complex isosurfaces. Interestingly, we found simple trilinear interpolation of stored coefficients sufficient. More complex interpolation schemes gave equivalent or even worse results, as shown in Fig. 6. Methods with larger kernels gave worse results due to the increased banding from interpolation of samples on widely different isosurfaces.

5 Results

We implemented our technique using several different approaches. We used two different rendering engines, an interactive raytracer running on parallel SGI machines and an interactive OpenGL visualization program running on a single PC.

To compute the values stored in the illumination texture, we used a parallelized Monte Carlo pathtracer based on the interactive raytracer. This prototype implementation uses unoptimized naive pathtracing. Combining this with our interactive raytracer allows lazy computation of the illumination texture. We did not implement lazy computation using our OpenGL application, though recent work [32] shows a similar approach may be feasible.

Our parallel implementation runs on an SGI Origin 3800 with 64 600 MHz R14000 processors. This is a shared memory machine with 32 GB of memory, allowing for easy access to large volume data sets and illumination textures. While large SGIs are uncommon, users generating and interactively displaying large volume data sets typically have access to similarly powerful machines (or clusters [33]) that could apply our technique. Our OpenGL implementation runs on a Dell Precision 450 with 1 GB memory and an Intel Xeon at 2.66 GHz. The graphics card is a GeForce FX 5900 with 256 MB memory. Because we use simple fragment shaders in our implementation, older cards will

TABLE 1
Illumination Computation Timings for Fig. 7

	100 samples	625 samples	2500 samples	10000 samples
Sphr. Harm. (lazy compute)	0.33 min	2.63 min	10.6 min	48.2 min
Sphr. Harm. (full texture)	8.47 min	52.8 min	210 min	853 min
Irradiance (lazy compute)	0.95 min	5.80 min	23.7 min	98.3 min
Irradiance (full texture)	18.2 min	113 min	450 min	1806 min
Pathtraced (single image)	0.73 min	4.51 min	18.0 min	72.1 min

Timings performed on 30 400 MHz R12000 CPUs.

work, but the increased graphics card memory facilitates visualizing bigger volumes.

Computing global illumination values for every texel in a volume can be quite expensive, whether computing simple irradiances or sets of spherical harmonic coefficients. Table 1 shows illumination computation timings for the $110 \times 208 \times 149$ engine block volume shown in Fig. 7. Times are shown for computing the entire texture as well as for the single views, computed at 900×900, shown in Fig. 7. In the case of the lazy computation, the timings represent the total time to compute illumination for a single image while running in interactive mode (i.e., the user could still interact with the object at around 2 frames per second). Our prototype uses naive, unoptimized Monte Carlo pathtracing for precomputation. More intelligent algorithms would significantly reduce the computation times in Table 1. Also note that 10,000 samples per texel are often unnecessary, even in complex environments. For the skull shown in Fig. 4, around 100 samples per texel suffices.

For volumes with few interesting surfaces such as the engine block, computing illumination on the fly may be preferential to a long precomputation, as global illumination samples reside near displayed isosurfaces. Samples elsewhere can remain uncomputed. At a 512×512 resolution, lazily computing a single irradiance for each visible sample takes a few seconds per viewpoint using 60 CPUs. Densely sampled illumination textures and complex environmental lighting require longer computations, as shown in Table 1. In either case, lazy computation maintains interactivity, so viewpoint and isovalue can be changed during computation.

While precomputation is slow, it need only be done once. Using the resulting illumination is simple and quicker than most lighting models used for visualization. Table 2 compares framerates from the scene in Fig. 7 using our raytraced-based renderer. The raytraced framerates are computed using 30 or 60 600 MHz R14000 CPUs on 512×512 images. Using either spherical harmonic coefficients or a single irradiance sample is faster than simple shading with hard shadows. Yet, both these techniques include shadows along with other global illumination effects. Using

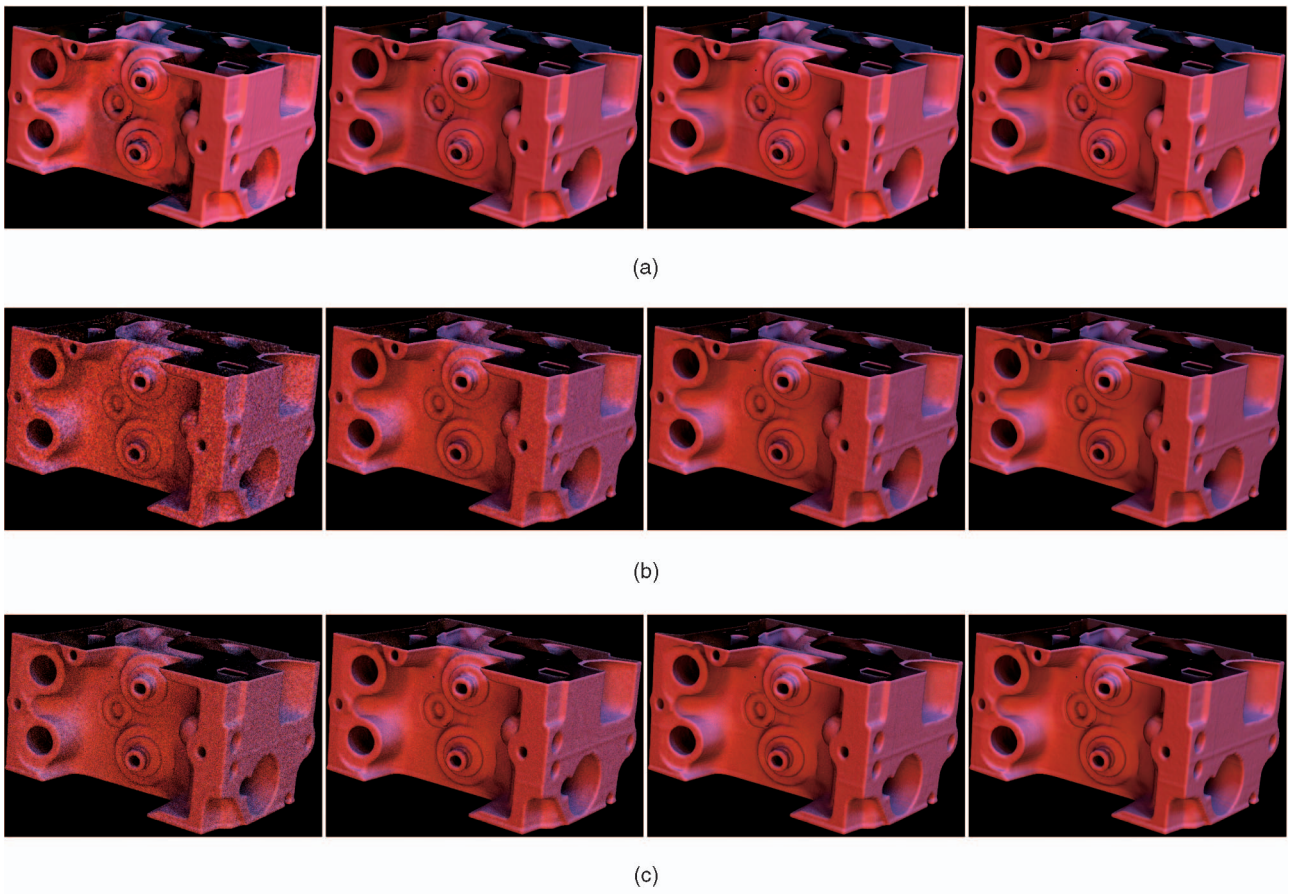

Fig. 7. The engine block illuminated by the Grace Cathedral lightprobe. Spherical harmonic samples (a) converge faster than Monte Carlo irradiance samples (b) or Monte Carlo pathtracing (c) due to the filtered low frequency environment. From left to right: 100, 625, 2,500, and 10,000 samples.

TABLE 2
Comparison of Timings on 512×512 Images and Memory Consumption Shows Times from the Raytracer on 30 and 60 CPUs and the OpenGL Implementation

Material	Frames per second (30 CPUs)	Frames per second (60 CPUs)	Frames per second (FX 5900)	Extra memory used
Diffuse (no shadows)	17.0	33.1	10-50	0
Diffuse (with shadows)	8.75	17.3	N/A	0
Phong (with shadows)	8.60	17.0	N/A	0
Irradiance	15.6	30.5	10-50	9.75 MB
Spherical harmonics	11.7	21.7	N/A	975 MB

Isosurface quality determines hardware rendering speed.

the OpenGL implementation, isosurface rendering is the bottleneck, as the global illumination is a simple texture lookup. Framerates range from 10 to 50 frames per second, depending on the quality of the isosurface rendered.

Table 2 also shows the memory overhead for our technique. The irradiance sample requires one RGB triplet per texel, which we store in three bytes. Using the same resolution as the volume data set requires as much as three times more memory. Our spherical harmonic representation uses no compression, so a fifth order representation uses 25 floating-point coefficients per channel, or two orders of magnitude more memory. Using compression techniques from Sloan et al. [17] would help reduce memory usage.

Simulation data, like the Richtmyer-Meshkov instability data set shown in Fig. 8 and Fig. 9, also benefits from global illumination. Often such data is confusing, so shadows and diffuse interreflections can give a sense of scale and depth lacking in Phong and Lambertian renderings. Fig. 8 and Fig. 9 compare our technique to vicinity shading, Phong, and Lambertian with and without shadows. Our Phong and Lambertian images use a varying ambient component based on the surface normal. We also compare to a local approach which adds distance information using OpenGL-style fog.

While vicinity shading provides better results than Phong or Lambertian models, incident illumination must be constant over the environment, such as on a cloudy day. Vicinity shading turns out to be a special case of our full global illumination solution. By ignoring diffuse bounces and insisting on constant illumination, we get identical results (as seen in Fig. 10). While vicinity shading shades concavities darker than unoccluded regions, recent studies show humans use more than a "darker-means-deeper" perceptual metric to determine shape [34]. By allowing interreflections between surfaces and more complex illumination, our approach adds additional lighting effects which may help users perceive shape. However, if shadows or other illumination effects inhibit perception for a particular data set, they can be removed from our illumination computations.

We get comparable results to Monte Carlo pathtracing, particularly for our irradiance texture. As we compute each irradiance texel using pathtracing, the differences seen in Fig. 11 result from the issues discussed in Sections 3 and 4.

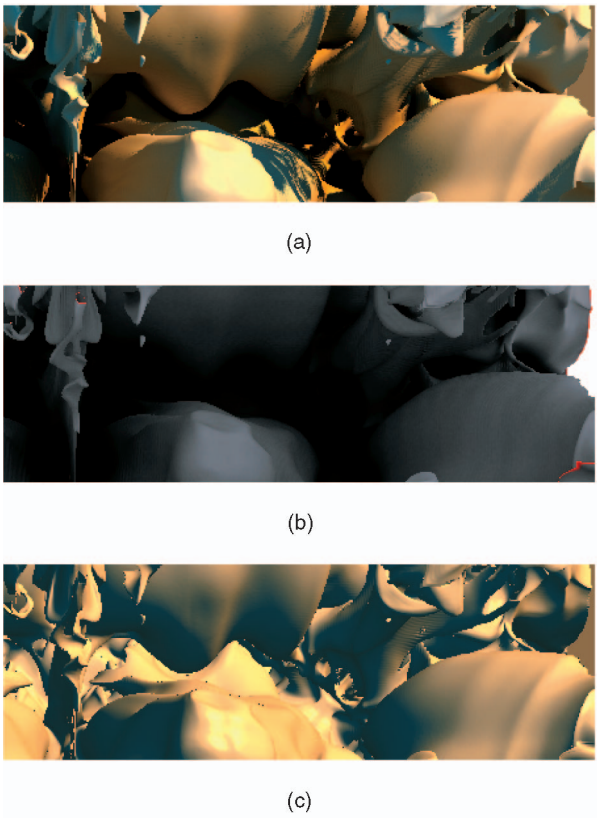

Fig. 8. An enlarged portion of the Richtmyer-Meshkov data set shown in Fig. 9. These images enlarge a crevice in the upper right corner of images from the right column using (a) our approach, (b) vicinity shading, and (c) Phong with varying ambient component.

For the spherical harmonic representation, our illumination is smoother and a bit darker due to the loss of detail from small, bright light sources. The illumination intensity varies slightly depending on the sampling of both the environment and material radiance transfer functions. Fig. 7 compares the convergence of illumination using a fifth order SH basis, single irradiance values, and per-pixel pathtracing. As expected, the SH representation converges significantly faster than pathtracing, and the irradiance texture converges at roughly the same rate, though the noise is blurred by the trilinear interpolation.

One last consideration when using our technique is what resolution illumination texture gives the best results. Fig. 12 compares the Visible Female head with three different resolutions. We found that using roughly the same resolution for the illumination and the data gave reasonable results for all our examples. In regions where isosurfaces vary significantly, sampling more finely may be desirable. For instance, illumination texels near the bone isosurface from Fig. 12 fall on relatively distant isosurfaces giving rise to more banding artifacts. Surfaces like the skin change slowly with changing isovalue, so a less dense illumination texture suffices.

6 Conclusion and Future Work

This paper introduces a method for precomputing and interactively rendering global illumination for surfaces from volume data sets. By storing illumination data in a 3D texture like the underlying volumetric data, interpolation between

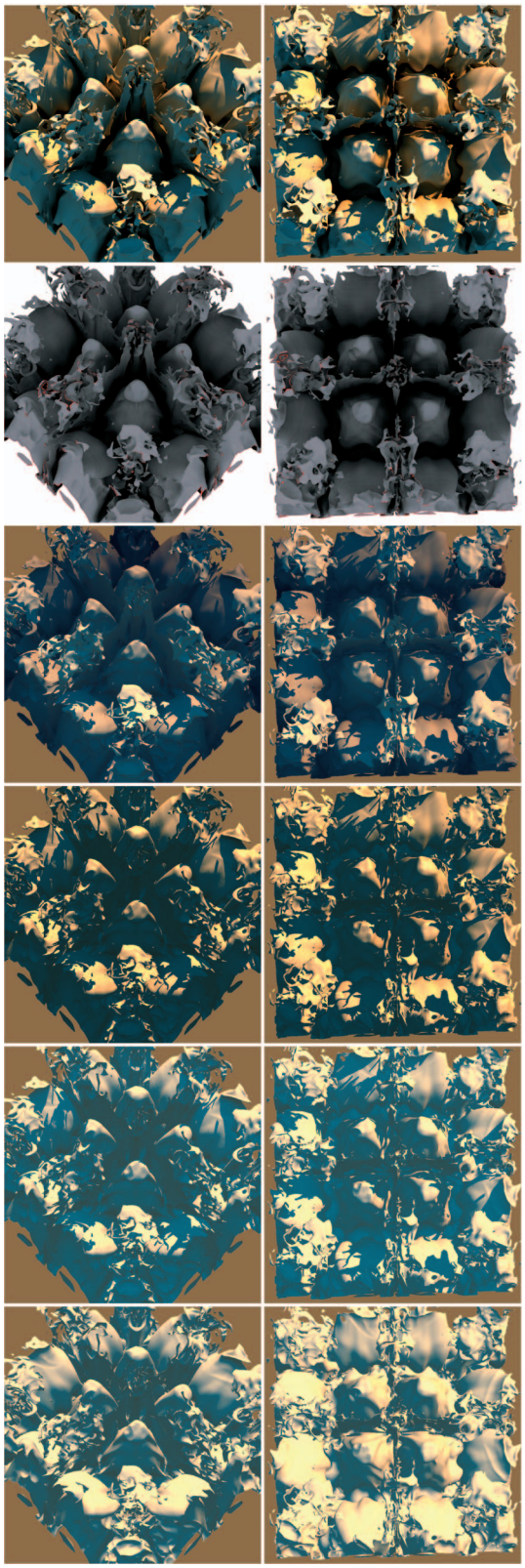

Fig. 9. Views from a Richtmyer-Meshkov instability simulation: (top to bottom) our technique, vicinity shading, Lambertian with fog, Phong with varying ambient, Lambertian with varying ambient, and Lambertian without shadows.

texels provides plausible global illumination at speeds faster than illumination models commonly used for visualization today. We have demonstrated that our approach generates high quality global illumination on dynamically changeable surfaces extracted from a volume, running on either GPU-based visualization tools or interactive raytracers. Combining the technique with a spherical harmonic representation allows dynamic environmental lighting.

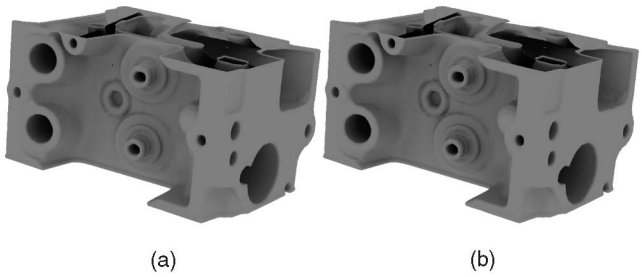

Fig. 10. Our technique (a) and vicinity shading (b) with 625 samples per voxel.

A number of issues warrant future examination. For instance, more complex material types can be represented using spherical harmonics at the cost of additional storage requirements. Due to the large number of samples required for an entire volume, this may not be feasible without more aggressive compression than that covered in Sloan et al. [17]. In some parts of a volume, the isosurfaces change slowly and smoothly requiring less dense samples. Yet, other regions demand extra samples to avoid artifacts. This suggests a hierarchical approach could prove helpful to reduce memory consumption. Future investigations could focus on when such hierarchies are useful and what representations allow quick access for rendering using GPU based renderers. Our illumination samples are currently computed either in advance or lazily using an interactive raytracer. Recent work [32] has discussed raytracing on graphics hardware. Such techniques may extend to allow computation of illumination samples on GPUs so expensive hardware is not required for interactive lazy computation. Finally, a user study investigating when data sets benefit from more complex illumination should prove interesting.

Appendix

Spherical Harmonics

Spherical harmonics provide an orthogonal basis for functions defined on a unit sphere. As global illumination deals with incident and exitant energies over spheres and hemispheres, spherical harmonics are a natural basis for representing these functions. Any function $f(\theta, \phi)$ over the unit sphere can be represented in terms of the complex spherical harmonic basis $\mathbb{Y}_l^m(\theta, \phi)$:

$$f(\theta, \phi) \equiv \sum_{l=0}^{\infty} \sum_{m=-l}^{l} A_l^m \mathbb{Y}_l^m(\theta, \phi), \tag{10}$$

where A_l^m are the spherical harmonic coefficients. Often, it is convenient to use the real-valued spherical harmonic basis functions $\mathbb{Y}_l^{m^c}$ and $\mathbb{Y}_l^{m^s}$, in which case $f(\theta, \phi)$ can be written as a generalized Fourier series:

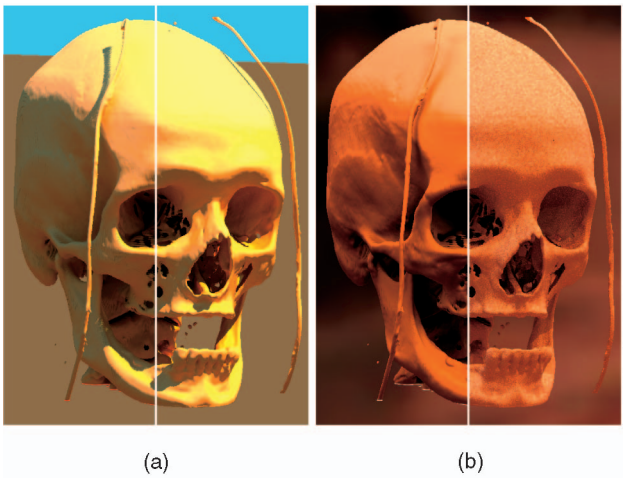

Fig. 11. Our technique (left half of each image) versus Monte Carlo pathtracing with 10,000 samples per pixel (right half of images). (a) Compares our irradiance texture to pathtracing and (b) compares a fifth order spherical harmonic representation to pathtracing.

$$f(\theta,\phi) \equiv \sum_{l=0}^{\infty}\sum_{m=0}^{l}\left[C_l^m \mathbb{Y}_l^{m^c}(\theta,\phi) + S_l^m \mathbb{Y}_l^{m^s}(\theta,\phi)\right]. \tag{11}$$

Often a single symbol $Y_l^m(\theta,\phi)$ is used to represent the real spherical harmonic functions, instead of the pair of symbols $\mathbb{Y}_l^{m^c}$ and $\mathbb{Y}_l^{m^s}$. In this case, functions f are represented as $f(\theta,\phi) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} C_l^m Y_l^m(\theta,\phi)$, where:

$$Y_l^m(\theta,\phi) = \begin{cases} \sqrt{2}N_l^m P_l^m(\cos\theta)\cos(m\phi) & \text{if } m > 0 \\ N_l^m P_l^0(\cos\theta) & \text{if } m = 0 \\ \sqrt{2}N_l^m P_l^m(\cos\theta)\sin(|m|\phi) & \text{if } m < 0. \end{cases} \tag{12}$$

Here, $P_l^m(x)$ are the associated Legendre functions and N_l^m is a normalization factor $\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$ chosen so that:

$$\int_0^{2\pi}\int_0^{\pi} Y_{l_1}^{m_1}(\theta,\phi)\overline{Y_{l_2}^{m_2}(\theta,\phi)}\sin\theta d\theta d\phi = \delta_{m_1,m_2}\delta_{l_1,l_2}. \tag{13}$$

Generally, a function is represented by a finite, nth-order approximation of the form:

$$f(\theta,\phi) \approx \sum_{l=0}^{n-1}\sum_{m=-l}^{l} C_l^m Y_l^m(\theta,\phi), \tag{14}$$

which allows representation of low frequency functions with a few C_l^m coefficients instead of a more complex form, like an environment map. However, this approximation leads to ringing and sharp discontinuities, especially for low order SH representations.

The property of SH representations that proves most useful for global illumination applications is the fact that integration of two functions $f(\theta,\phi)$ and $g(\theta,\phi)$ simplifies to a dot product of their spherical harmonic coefficients. In other words, if f has SH coefficients a_l^m and g has coefficients b_l^m then:

$$\begin{aligned}\int_{\theta,\phi} f(\theta,\phi)g(\theta,\phi)d\theta d\phi &= \sum_{l=0}^{\infty}\sum_{m=-l}^{l} a_l^m b_l^m \\ &\approx \sum_{l=0}^{n-1}\sum_{m=-l}^{l} a_l^m b_l^m = \sum_{i=0}^{n^2} a_i b_i,\end{aligned} \tag{15}$$

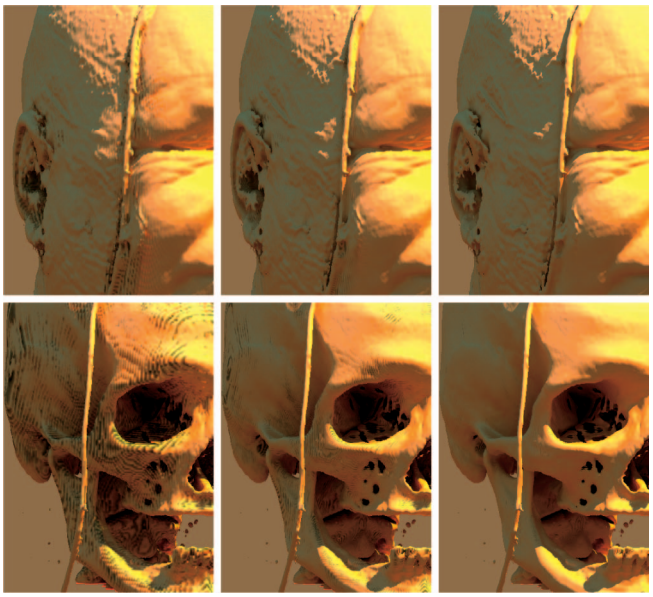

Fig. 12. Illumination texture of (left to right) 1/8, 1, and 8 times the resolution of the head data set. Due to the high variation in isovalues near the bone isosurface, a denser illumination sampling is needed to avoid banding artifacts.

where, for ease of use, coefficients are indexed with a single dimensional array a_i instead of the two-dimensional array a_l^m. In this case, $i = l(l+1)+m$ and the sum in (15) simplifies to a summation over i. Similarly, Y_i is used for Y_l^m.

Another useful property of spherical harmonics is that coefficients can be rotated via matrix transformations either before or after projection. Since projection of an environment map can be expensive, this property allows dynamic rotation of the illumination without reprojecting every frame. Furthermore, coefficients in different spherical harmonic bands (different l values) do not interact during rotation, so a spherical harmonic rotation matrix R has the form:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & & & & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & & R^1 & & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & & & & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & & & & & & \cdots \\ 0 & 0 & 0 & 0 & & & & & & \cdots \\ 0 & 0 & 0 & 0 & & & R^2 & & & \cdots \\ 0 & 0 & 0 & 0 & & & & & & \cdots \\ 0 & 0 & 0 & 0 & & & & & & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where R^i are $(2i+1)\times(2i+1)$ matrices which rotate the coefficients of the ith band. Given a rotation $R_{\mathbb{S}}$ on the sphere, the matrix elements of R are:

$$R_{s,t} = \int Y_s(R_{\mathbb{S}}(\vec{\omega}\,'))Y_t(\vec{\omega}\,')d\vec{\omega}\,'. \tag{16}$$

While these matrices are complicated, they can be computed quickly in a couple of ways. One common way of representing rotations is a ZYZ Euler decomposition. Rotations of SH coefficients around the z-axis prove simple, so often the rotation around the y-axis is further decomposed into a rotation of $\frac{\pi}{2}$ about the x-axis, a rotation about the z-axis, and a rotation of $-\frac{\pi}{2}$ about the x-axis. Green [30] gives the

TABLE 3
Definitions of Coefficients u, v, and w and the Functions U, V, W, and P

Coefficient	for $\lvert t\rvert < i$	for $\lvert t\rvert = i$
$u^i_{s,t}$:	$\sqrt{\frac{(i+s)(i-s)}{(i+t)(i-t)}}$	$\sqrt{\frac{(i+s)(i-s)}{2i(2i-1)}}$
$v^i_{s,t}$:	$\frac{1}{2}\sqrt{\frac{(1+\delta_{s,0})(i+\lvert s\rvert-1)(i+\lvert s\rvert)}{(i+t)(i-t)}}(1-2\delta_{s,0})$	$\frac{1}{2}\sqrt{\frac{(1+\delta_{s,0})(i+\lvert s\rvert-1)(i+\lvert s\rvert)}{2i(2i-1)}}(1-2\delta_{s,0})$
$w^i_{s,t}$:	$-\frac{1}{2}\sqrt{\frac{(i-\lvert s\rvert-1)(i-\lvert s\rvert)}{(i+t)(i-t)}}(1-\delta_{s,0})$	$-\frac{1}{2}\sqrt{\frac{(i-\lvert s\rvert-1)(i-\lvert s\rvert)}{2i(2i-1)}}(1-\delta_{s,0})$

(a)

Function	for $s = 0$	for $s > 0$	for $s < 0$
$U^i_{s,t}$:	${}_0P^i_{0,t}$	${}_0P^i_{s,t}$	${}_0P^i_{s,t}$
$V^i_{s,t}$:	${}_1P^i_{1,t} + {}_{-1}P^i_{-1,t}$	${}_1P^i_{s-1,t}\sqrt{1+\delta_{s,1}} - {}_{-1}P^i_{-s+1,t}(1-\delta_{s,1})$	${}_1P^i_{s+1,t}(1-\delta_{s,-1}) + {}_{-1}P^i_{-s+1,t}\sqrt{1+\delta_{s,-1}}$
$W^i_{s,t}$:	(N/A, as $w^i_{0,t} = 0$)	${}_1P^i_{s+1,t} + {}_{-1}P^i_{-s-1,t}$	${}_1P^i_{s-1,t} + {}_{-1}P^i_{-s+1,t}$

(b)

Function	for $\lvert t\rvert < i$	for $t = i$	for $t = -i$
${}_aP^i_{s,t}$:	$r_{a,0}R^{i-1}_{s,t}$	$r_{a,1}R^{i-1}_{s,i-1} - r_{a,-1}R^{i-1}_{s,-i+1}$	$r_{a,1}R^{i-1}_{s,-i+1} + r_{a,-1}R^{i-1}_{s,i-1}$

(c)

(a) Definitions of numerical coefficients $u^i_{s,t}$, $v^i_{s,t}$, and $w^i_{s,t}$. (b) Definitions of the functions $U^i_{s,t}$, $V^i_{s,t}$, and $W^i_{s,t}$. (c) Definitions of the function ${}_aP^i_{s,t}$.

matrices for rotation around the z-axis and the $\pm\frac{\pi}{2}$ rotations about the x-axis.

Recently, Ivanic and Ruedenberg [35], [36] introduced a recursive technique to generate an arbitrary rotation matrix, which allows rotation using a single matrix instead of a composition of five separate matrices. Given an ordinary 3×3 rotation matrix $R_{\mathbb{S}}$ defined as:

$$R_{\mathbb{S}} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,-1} & r_{1,0} \\ r_{-1,1} & r_{-1,-1} & r_{-1,0} \\ r_{0,1} & r_{0,-1} & r_{0,0} \end{bmatrix},$$

then for an nth order spherical harmonic (i.e., $0 \le i < n$ and $-i \le s, t \le i$):

$$R^i_{s,t} = u^i_{s,t}U^i_{s,t} + v^i_{s,t}V^i_{s,t} + w^i_{s,t}W^i_{s,t}. \tag{17}$$

Here, s and t are indices ranging over elements of the $(2i+1) \times (2i+1)$ rotation matrix for the ith band of spherical harmonic coefficients. The coefficients u, v, w and functions U, V, W, P appear in Table 3 and define the recurrence relation. Note this table is from the original version of the Ivanic and Ruedenberg paper [35], which contains erroneous equations but correct tables.

Acknowledgments

The authors would like to thank David Banks for initially suggesting the idea of caching illumination for isosurfaces during a 1999 visit to Utah. Also deserving of thanks are numerous anonymous reviewers who shared valuable insights and suggestions for improving their work. This material is based on work supported by the US National Science Foundation under Grants: 9977218 and 9978099.

References

[1] W. Schroeder, K. Martin, and W. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics.* Prentice Hall, 2003.
[2] A.E. Kaufman, "Volume Visualization in Medicine," *Handbook of Medical Imaging,* Academic Press, pp. 713-730, 2000.
[3] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley, "Interactive Ray Tracing for Volume Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 3, pp. 287-296, July-Sept. 1999.
[4] G. Miller, "Efficient Algorithms for Local and Global Accessibility Shading," *Proc. ACM SIGGRAPH '94,* pp. 319-326, 1994.
[5] S. Zhukov, A. Iones, and G. Kronin, "An Ambient Light Illumination Model," *Proc. Eurographics Rendering Workshop,* pp. 45-56, June 1998.
[6] M. Pharr and S. Green, *GPU Gems,* Addison Wesley, chapter on ambient occlusion, pp. 279-292, 2004.
[7] J. Stewart, "Vicinity Shading for Enhanced Perception of Volumetric Data," *Proc. Visualization Conf.,* pp. 355-362, 2003.
[8] G. Greger, P. Shirley, P.M. Hubbard, and D.P. Greenberg, "The Irradiance Volume," *IEEE Computer Graphics and Applications,* vol. 18, no. 2, pp. 32-43, Mar.-Apr. 1998.
[9] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 527-536, 2002.

[10] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH '87,* pp. 163-169, 1987.
[11] B. Lichtenbelt, R. Crane, and S. Naqvi, *Introduction to Volume Rendering,* first ed. Prentice Hall, 1998.
[12] B.T. Phong, "Illumination for Computer Generated Images," *Comm. ACM,* vol. 18, pp. 311-317, 1975.
[13] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 99-108, Apr.-June 1995.
[14] L.M. Sobierajski and A.E. Kaufman, "Volumetric Ray Tracing," *Proc. Symp. Volume Visualization,* pp. 11-18, 1994.
[15] U. Behrens and R. Ratering, "Adding Shadows to a Texture-Based Volume Renderer," *Proc. IEEE Symp. Volume Visualization,* pp. 39-46, 1998.
[16] R. Ng, R. Ramamoorthi, and P. Hanrahan, "All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 376-381, 2003.
[17] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder, "Clustered Principal Components for Precomputed Radiance Transfer," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 382-391, 2003.
[18] 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.
[19] X. Granier and G. Drettakis, "Incremental Updates for Rapid Glossy Global Illumination," *Computer Graphics Forum,* vol. 20, no. 3, pp. 268-277, 2001.
[20] D. Forsyth, C. Yang, and K. Teo, "Efficient Radiosity in Dynamic Environments," *Proc. Eurographics Rendering Workshop,* pp. 313-323, 1994.
[21] P. Tole, F. Pellacini, B. Walter, and D. Greenberg, "Interactive Global Illumination in Dynamic Scenes," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 537-546, 2002.
[22] I. Wald, C. Benthin, and P. Slusallek, "Interactive Global Illumination in Complex Highly Occluded Environments," *Proc. Eurographics Symp. Rendering,* pp. 74-81, 2003.
[23] K. Beason, J. Grant, D. Banks, B. Futch, and M.Y. Hussaini, "Pre-Computed Illumination for Isosurfaces," *Proc. Conf. Visualization and Data Analysis,* to appear, 2006.
[24] G.J. Ward, F.M. Rubinstein, and R.D. Clear, "A Ray Tracing Solution for Diffuse Interreflection," *Proc. ACM SIGGRAPH '88,* pp. 85-92, 1988.
[25] K. Bala, J. Dorsey, and S. Teller, "Radiance Interpolants for Accelerated Bounded-Error Ray Tracing," *ACM Trans. Graphics,* vol. 18, no. 3, pp. 213-256, 1999.
[26] R. Ramamoorthi and P. Hanrahan, "An Efficient Representation for Irradiance Environment Maps," *Proc. ACM SIGGRAPH,* pp. 497-500, 2001.
[27] W.T. Reeves, D.H. Salesin, and R.L. Cook, "Rendering Antialiased Shadows with Depth Maps," *Proc. ACM SIGGRAPH '87,* pp. 283-291, 1987.
[28] J.T. Kajiya, "The Rendering Equation," *Proc. ACM SIGGRAPH '86,* vol. 20, no. 4, pp. 143-150, 1986.
[29] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan, "Interactive Ray Tracing for Isosurface Rendering," *Proc. Visualization Conf. '98,* pp. 233-238, Oct. 1998.
[30] R. Green, "Spherical Harmonic Lighting: The Gritty Details," *Proc. Archives of the Game Developers Conf.,* Mar. 2003.
[31] S.R. Marschner and R.J. Lobb, "An Evaluation of Reconstruction Filter for Volume Rendering," *Proc. Visualization Conf.,* pp. 100-107, Oct. 1994.
[32] T.J. Purcell, I. Buck, W.R. Mark, and P. Hanrahan, "Ray Tracing on Programmable Graphics Hardware," *ACM Trans. Graphics,* vol. 21, no. 4, pp. 703-712, 2002.
[33] D.E. Demarle, S. Parker, M. Hartner, C. Gribble, and C. Hansen, "Distributed Interactive Ray Tracing for Large Volume Visualization," *Proc. Symp. Parallel and Large-Data Visualization and Graphics,* pp. 87-94, 2003.
[34] M.S. Langer and H.H. Bülthoff, "Depth Discrimination from Shading under Diffuse Lighting," *Perception,* vol. 29, pp. 649-660, 2000.
[35] J. Ivanic and K. Ruedenberg, "Rotation Matrices for Real Spherical Harmonics, Direct Determination by Recursion," *J. Physical Chemistry A,* vol. 100, no. 15, pp. 6342-6347, 1996.
[36] "Additions and Corrections: Rotation Matrices for Real Spherical Harmonics," *J. Physical Chemistry A,* vol. 102, no. 45, pp. 9099-9100, 1998.

Chris Wyman received the PhD degree in computer science from the University of Utah in 2004. He received the BS degree in mathematics and computer science from the University of Minnesota in 1999. He is an assistant professor in the Department of Computer Science at the University of Iowa. His professional interests focus on interactive global Illumination, but also extend to other interactive and realistic rendering problems and visualization. He is a member of the IEEE.

Steven Parker received the BS degree in electrical engineering from the University of Oklahoma in 1992, and the PhD degree from the University Utah in 1999. He is a research assistant professor in the School of Computing and Scientific Computing and Imaging (SCI) Institute at the University of Utah. His research focuses on problem solving environments, which tie together scientific computing, scientific visualization, and computer graphics. He is the principal architect of the SCIRun Problem-Solving Environment, which formed the core of his PhD dissertation, the manta interactive ray tracing system, and is currently the chief architect of Uintah, a software system designed to simulate accidental fires and explosions using thousands of processors. He was a recipient of the Computational Science Graduate Fellowship from the Department of Energy. He is a member of the IEEE.

Peter Shirley received the BA degree in physics from Reed College and the PhD degree in computer science from the University of Illinois at Urbana-Champaign. He is a professor in the School of Computing at the University of Utah. He spent four years as an assistant professor at Indiana University and two years as a visiting assistant professor at the Cornell Program of Computer Graphics before moving to Utah. His professional interests include interactive and realistic rendering, statistical computing, visualization, and immersive environments.

Charles Hansen received the BS degree in computer science from Memphis State University in 1981 and the PhD degree in computer science from the University of Utah in 1987. He is a professor of computer science at the University of Utah. From 1997 to 1999, he was a research associate professor of computer science at Utah. From 1989 to 1997, he was a technical staff member in the Advanced Computing Laboratory (ACL). He was a Bourse de Chateaubriand postdoctorate fellow at INRIA, Rocquencourt France, in 1987 and 1988. His research interests include large-scale scientific visualization and computer graphics. He is a senior member of the IEEE.