Interactive Display of Isosurfaces with Global Illumination

Chris Wyman, Steven Parker, Peter Shirley, Charles Hansen

UUCS-04-012

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

July 22, 2004

Abstract

In many applications, volumetric datasets 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 viewpoint and isovalue.

Interactive Display of Isosurfaces with Global Illumination

Chris Wyman Steven Parker Peter Shirley Charles Hansen

University of Utah

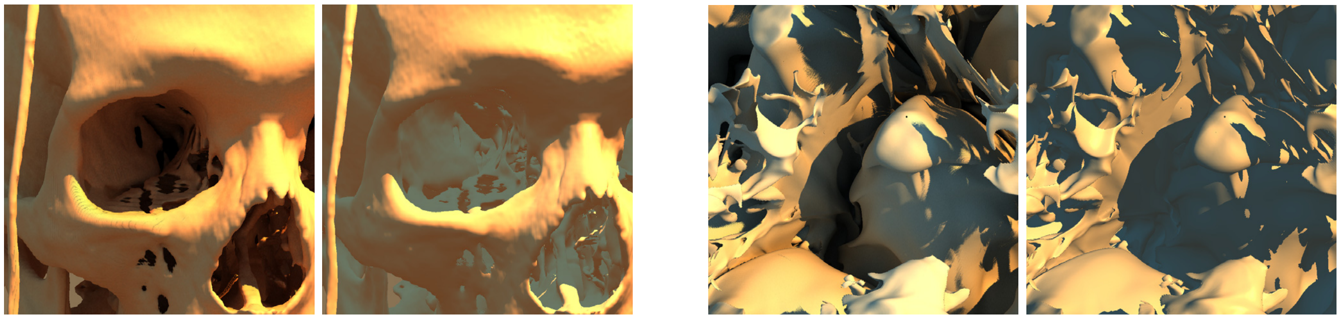

Figure 1: The left image in each pair shows our approach, the right shows Phong with shadows. Note the improvement in regions dominated by indirect lighting, particularly in the eye sockets (left) and the concavities in the simulation data (right).

Abstract

In many applications, volumetric datasets 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 viewpoint and isovalue.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: path tracing, isosurface, visualization, rendering, global illumination

1 Introduction

Isosurfaces, also known as implicit surfaces or level sets, are widely used in computer graphics and scientific visualization, whether to help model complex objects [21] or to reveal structure of scalar-valued functions and medical imaging data [6]. Because both computational and acquired datasets tend to be large, only recently has interactive display and manipulation of isosurfaces become feasible for full-resolution datasets [15]. 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. Figure 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* [13] and *obscurance shading* [27]. More recently, direct lighting from a uniform extended source has been applied to illuminating isosurfaces extracted from volumes to good effect [25].

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 stores the necessary global illumination for all isosurfaces. For dynamic lights and complex materials, multiple values are used per texel, as per Sloan et al. [23]. 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 datasets 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 datasets 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 either a rectilinear or tetrahedral lattice.

Given a sampled dataset $\rho(\vec{x}_i)$, 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 a polygonal representation [15, 9]. Once an isosurface has been identified, standard illumination models can be applied to help visualize the data.

Commonly, extracted isosurfaces are shaded using the Phong model [17] 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 [12], 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 [24] apply global illumination to volume datasets, but they shade at runtime, so few effects can be incorporated while maintaining interactivity.

Several techniques have been proposed to interactively shade volumes with global lighting. The Irradiance Volume [4] 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 [23, 22, 14] 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 datasets. Kniss et al. [7] 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 [25] 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.

3 Overview

A brute-force approach to globally illuminating an isosurface would compute illumination at every point visible from the eye. Figure 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 [26] are usually preferable. Many existing techniques cache radiance [1], irradiance [4], or more complex transfer functions [19, 23, 14] 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 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. Figure 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.

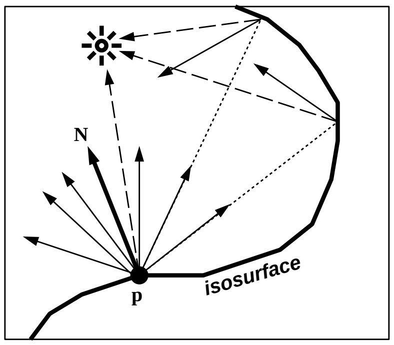

Figure 2: Computing the irradiance at point $\vec{p}$ involves sending a shadow ray and multiple reflection rays. Reflection rays either hit the background or another part of the isosurface, in which case rays are recursively generated.

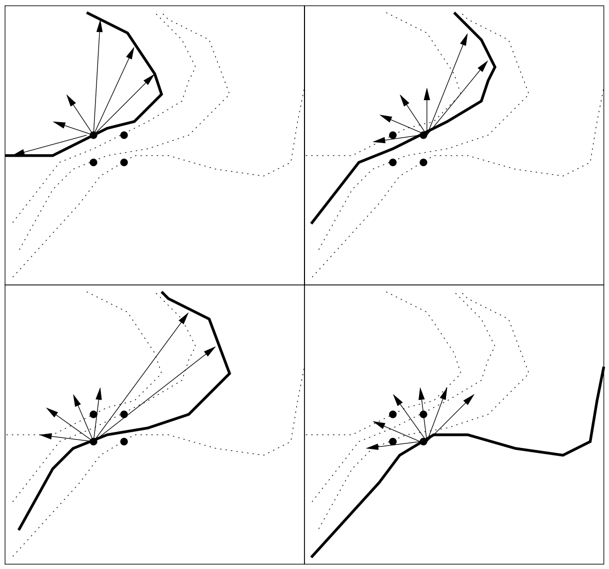

Figure 3: The global illumination at each texel t is computed using standard techniques based on the isosurface $I(\rho(\vec{x}_t))$ through the sample.

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 Figure 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 [20].

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 disap-

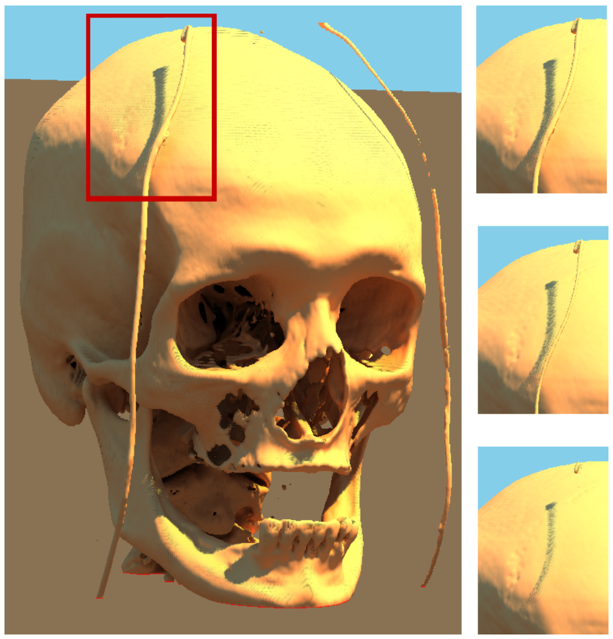

Figure 4: The Visible Female's skull globally illuminated using our technique. The right images show how the cord's shadow fades out with increasing isovalues.

pear. Examples of these effects can be seen in Figure 4. The scene is illuminated by a blue and brown environment 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 shadow 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 Figure 4 may still occur, they will be less noticeable.

The artifacts that occur with changing isovalues arise from our approximation of the non-linear 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 computation. In fact, our technique 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 computations as a preprocess before rendering. As this can require significant time and not all illumination data may be required, it is 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 [5]:

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

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, and f_r is the BRDF.

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:

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

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 Equation 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:

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

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 $\Omega$ 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 equation 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 [16]. Unfortunately, trilinear techniques generate noisy surfaces and normals, which can significantly impact the quality of the computed global illumination (see Figure 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 $4\times4\times4$ 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 Figure 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 dataset 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 Figure 5.

For more complex effects such as dynamic illumination or non-diffuse 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

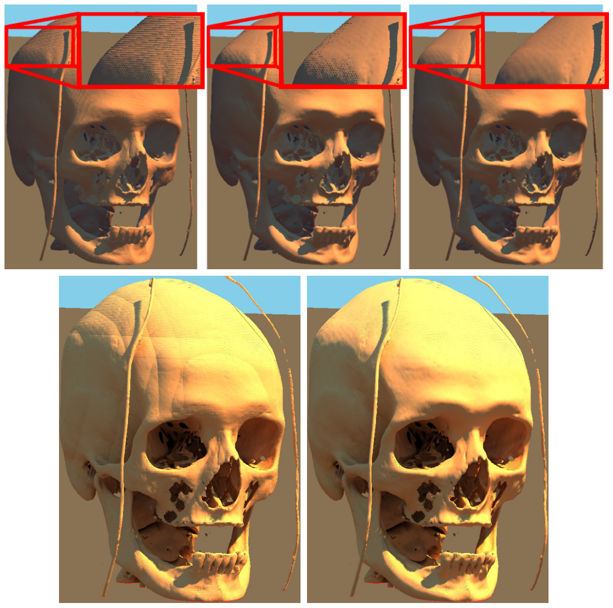
Figure 5: An isosurface from the Visible Female's head extracted using analytical intersection of the trilinear surface. Top: Direct illumination from a point light using (left) gradient normals, (center) tricubic B-spline smoothed normals, and (right) offset surface with smoothed normals. Bottom: Four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

to represent more complex illumination as spherical harmonics represent low frequency lighting efficiently. They allow for quick integration during rendering, using a simple dot product or matrix multiply operation, as well as dynamically changing illumination.

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:

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

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 whose coefficients can be computed numerically using Monte Carlo techniques. Green [3] clearly explains this process in great detail. We store the SH coefficients of this geometry term at each point in our illumination texture.

Similar SH values can be computed and stored at voxels for more complex materials including glossy or transparent effects, as described in Sloan et al. [23, 22]. Additionally other bases, like wavelets, could be used to represent the global illumination in our volume, especially if higher-frequency effects are desired.

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.

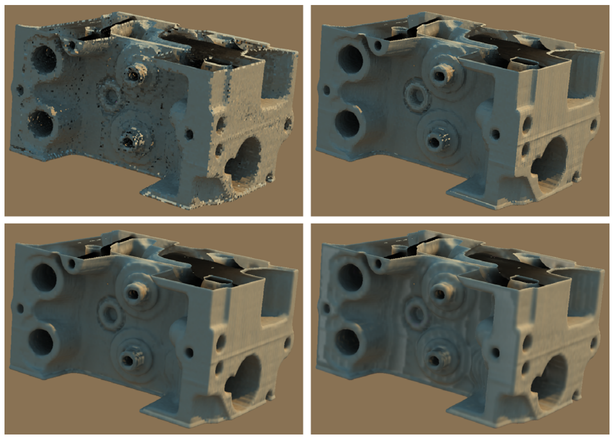
Figure 6: Engine block illuminated using (top left) the illumination sample with closest isovalue, (top right) the nearest illumination sample, (bottom left) a trilinearly interpolated value, or (bottom right) a value computed with a tricubic B-Spline kernel.

When using an irradiance texture, we simply use Equation 2 to compute the illumination based on the stored irradiance and the albedo. With the spherical harmonic representation, we interpolate the stored spherical harmonic geometry coefficients and perform a vector dot product with the environmental lighting coefficients.

We expected a higher order interpolation [11] 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 Figure 6. Methods with larger kernels generally gave worse results due to the increased likelihood of interpolating samples from widely different isosurfaces.

5 Results

We implemented our technique using several different approaches. We used a Monte Carlo pathtracer to compute the illumination at texels throughout the volume. Utilizing the precomputations in our interactive raytracer allows dynamic changes to the visualized surface and illumination. We extended our interactive raytracer to compute illumination lazily, avoiding computations for surfaces never seen. Finally, we imported our illumination texture into an OpenGL visualization program to render global illumination interactively on a single PC.

Our parallel implementation runs on an SGI Origin 3800 with sixty-four 600 MHz R14000 processors. This is a shared memory machine with 32 GB of memory, allowing for easy access to large volume datasets and illumination textures. While large SGIs are uncommon, users generating and interactively displaying large volume datasets typically have access to similarly powerful machines (or clusters [2]) which 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 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 volume shown in Figure 7. Times are shown for computing the entire texture as well as for the single views shown in Figure 7. Our prototype uses naive, unoptimized

	100 samples	625 samples	2500 samples	10000 samples
Sphr. Harm. (single image)	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 (single image)	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

Table 1: Illumination computation timings for images from Figure 7. Timings performed on thirty 400 MHz R12000 CPUs.

Material	Frames per second (30 CPUs)	Frames per second (60 CPUs)	Extra memory used
Diffuse (no shadows)	17.0	33.1	0
Diffuse (with shadows)	8.75	17.3	0
Phong (with shadows)	8.60	17.0	0
Irradiance samples	15.6	30.5	9.75 MB
Spherical harmonics	11.7	21.7	975 MB

Table 2: Comparison of timings and memory consumption. Results use our raytraced implementation on thirty or sixty 600 MHz R14000 CPUs with 512×512 resolution for the scene in Figure 7.

Monte Carlo pathtracing for precomputation. More intelligent algorithms would significantly reduce these precomputation times.

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 the 512×512 resolution shown in the accompanying video, computing a single irradiance for each 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 using Phong and Lambertian materials with our technique. 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.

Table 2 also shows the memory overhead for our technique. The irradiance sample requires one RGB triplet per voxel, which we store in three bytes. Using the same resolution as the volume dataset 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. [22] would help reduce memory usage.

Simulation data, like the Richtmyer-Meshkov instability dataset

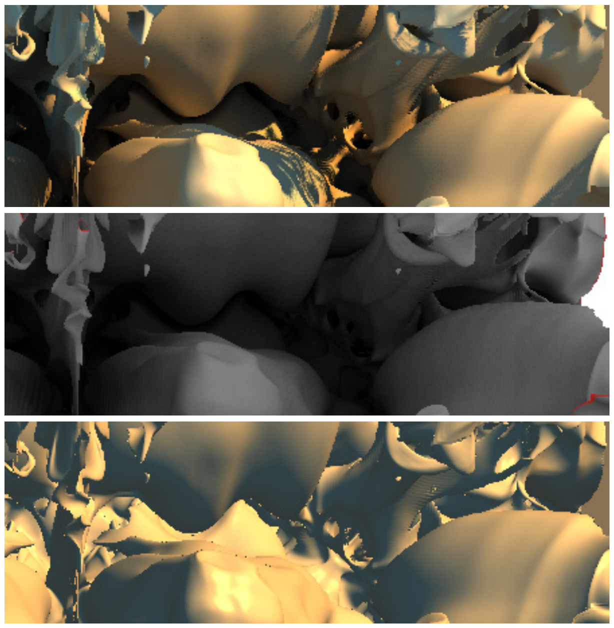

Figure 8: An enlarged portion of the Richtmyer-Meshkov dataset shown in Figure 9. These images enlarge a crevice in the upper right corner of images from the right column using (top) our approach, (center) vicinity shading, and (bottom) Phong with varying ambient component.

shown in Figures 8 and 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. Figures 8 and 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 Figure 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 in an image [8]. 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 dataset, 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 Figure 11 result from the issues discussed in Sections 3 and 4. For the spherical harmonic representation, our illumination is smoother and a bit darker. The illumination intensity varies slightly based on the SH sampling of the environment and material transfer functions, and our SH results in Figure 7 appear a bit brighter than the pathtraced results. Figure 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 interpola-

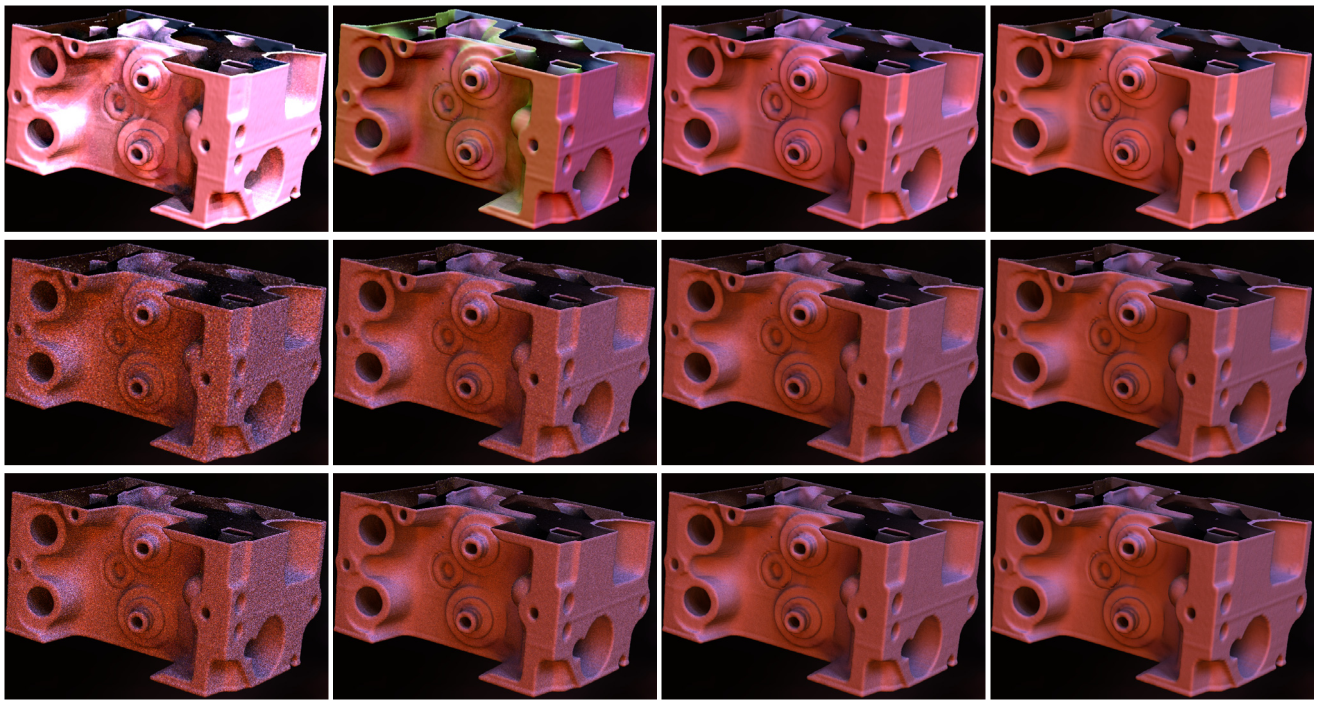

Figure 7: The engine block illuminated by the Grace Cathedral lightprobe. Spherical harmonic samples (top) converge faster than Monte Carlo irradiance samples (middle) or Monte Carlo pathtracing (bottom) due to the filtered low frequency environment. From left to right: 100, 625, 2500, and 10000 samples.

tion.

One last consideration when using our technique is what resolution illumination texture gives the best results. Figure 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 Figure 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 datasets. By storing illumination data in a 3D texture like the underlying volumetric data, interpolation between 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.

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. [22]. 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 [18] 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 datasets benefit from more complex illumination should prove interesting.

References

[1] Kavita Bala, Julie Dorsey, and Seth Teller. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Transactions on Graphics*, 18(3):213–256, 1999.

[2] David E. Demarle, Steven Parker, Mark Hartner, Christiaan Gribble, and Charles Hansen. Distributed interactive ray tracing for large volume visualization. In *Proceedings of the Symposium on Parallel and Large-Data Visualization and Graphics*, pages 87–94. ACM Press, 2003.

[3] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the Game Developers Conference*, March 2003.

[4] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, March-April 1998.

[5] James T. Kajiya. The rendering equation. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, volume 20, pages 143–150, 1986.

[6] Arie E. Kaufman. Volume visualization in medicine. In *Handbook of Medical Imaging*, pages 713–730. Academic Press, 2000.

[7] Joe Kniss, Simon Premoze, Charles Hansen, Peter Shirley, and Allan McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, April 2003.

[8] Michael S. Langer and Heinrich H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29:649–660, 2000.

[9] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi. *Introduction to Volume Rendering*. Prentice Hall, 1st edition, 1998.

[10] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics(Proceedings of ACM SIGGRAPH 87)*, volume 21, pages 163–169. ACM, 1987.

[11] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filter for volume rendering. In *Proceedings of Visualization '98*, pages 100–107, October 1994.

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

[13] Gavin Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of ACM SIGGRAPH 94*, pages 319–326. ACM Press/ACM SIGGRAPH, 1994.

[14] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics*, 22(3):376–381, 2003.

[15] Steven Parker, Michael Parker, Yarden Livnat, Peter-Pike Sloan, Charles Hansen, and Peter Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):287–296, July 1999.

[16] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings of Visualization '98*, pages 233–238, October 1998.

[17] Bui Thong Phong. Illumination for computer generated images. *Communications of the ACM*, 18:311–317, 1975.

[18] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions of Graphics*, 21(4):703–712, 2002.

[19] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH 2001*, pages 497–500. ACM Press/ACM SIGGRAPH, 2001.

[20] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, volume 21, pages 283–291. ACM, 1987.

[21] Will Schroeder, Ken Martin, and William Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 3rd edition, 2003.

[22] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, 22(3):382–391, 2003.

[23] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, 2002.

[24] Lisa M. Sobierajski and Arie E. Kaufman. Volumetric ray tracing. In *Proceedings of the Symposium on Volume Visualization*, pages 11–18. ACM Press, 1994.

[25] James Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of Visualization*, pages 355–362, 2003.

[26] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, volume 22, pages 85–92. ACM, 1988.

[27] Sergej Zhukov, Andrej Iones, and Grigorij Kronin. An ambient light illumination model. In *Eurographics Rendering Workshop*, pages 45–56, June 1998.

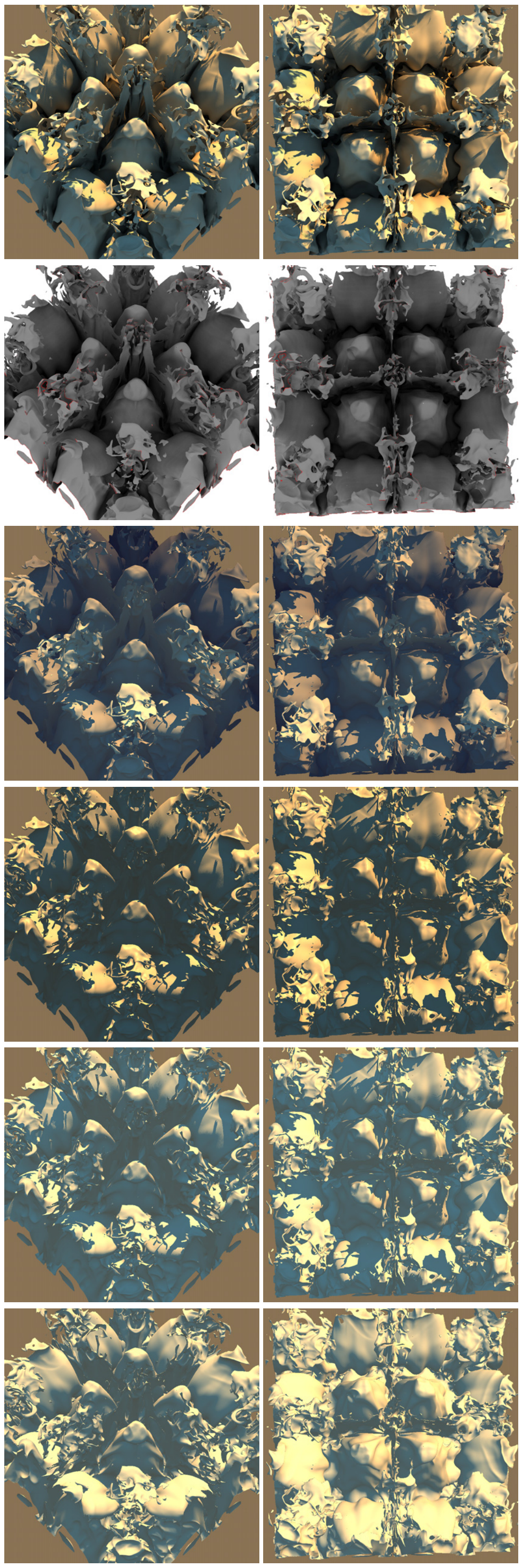

Figure 9: Views from a Richtmyer-Meshkov instability simulation: (top to bottom) our technique, vicinity shading, Lambertian with fog, Phong with varying ambient, Lambertian with varying ambient, and Lambertian without shadows.

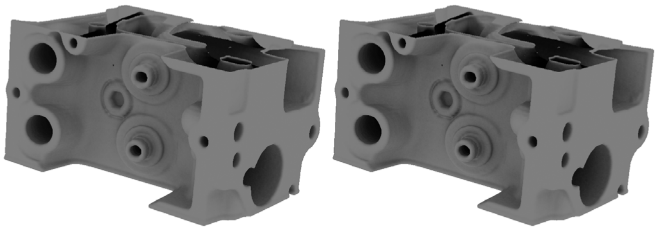

Figure 10: Our technique (left) and vicinity shading (right) with 625 samples per voxel.

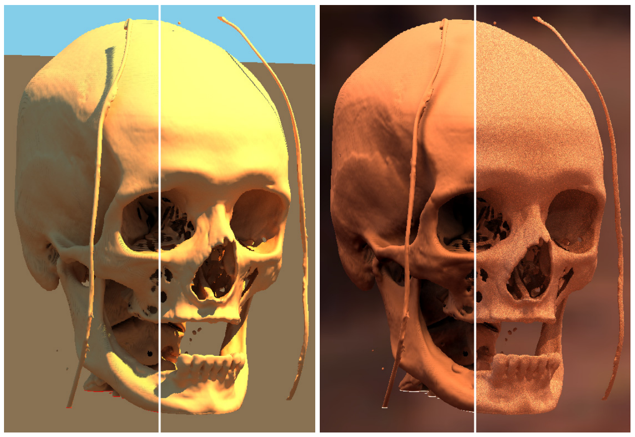

Figure 11: Our technique (left half of each image) versus Monte Carlo pathtracing with 10000 samples per pixel (right half of images). The left image compares our irradiance texture to pathtracing, the right image compares a fifth order spherical harmonic representation to pathtracing.

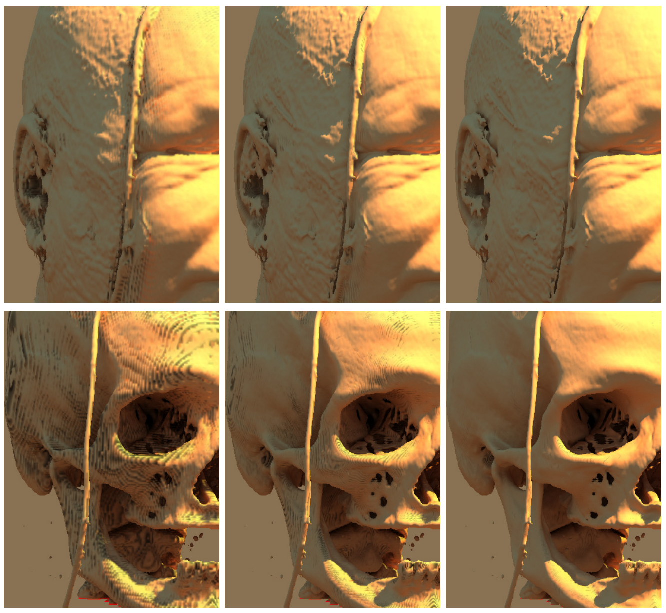

Figure 12: Illumination texture of (left to right) 1/8, 1, and 8 times the resolution of the head dataset. Due to the high variation in isovalues near the bone isosurface, a denser illumination sampling is needed to avoid banding artifacts.