Volumetric Global Illumination and Reconstruction via Energy Backprojection

Frank Dachille IX, Klaus Mueller, and Arie Kaufman
Center for Visual Computing (CVC)
and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

Volumetric energy backprojection captures the effects of myriad physical processes including global illumination and reconstruction. We present a method to perform efficient volumetric backprojection in software. We develop a new method for global illumination based on iterated volumetric backprojection. We demonstrate how computed tomography and visible light reconstruction can be implemented using volumetric backprojection. Our new form of opaque reconstruction is insensitive to shading and includes the partial volume effect. Finally, we suggest small modifications to volume rendering hardware which permits efficient, scalable volumetric backprojection.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture; I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—; I.4.5 [Image Processing and Computer Vision]: Reconstruction;

Keywords: Global illumination, radiosity, radiative transport, volume rendering, reconstruction, opaque reconstruction

1 Introduction

Radiative transport, for example, illumination, most often occurs distributed simultaneously throughout 3D space, traveling in linear paths and interacting locally with intervening media. Many natural phenomena such as radiative transport can be characterized by volumetric backprojection. Volumetric backprojection is the name given to the class of operations which project and distribute energy through a discrete 3D grid. The intent of this paper is to investigate a variety of uses for volumetric backprojection, examine the relative efficiency of computation schemes, and suggest a simple hardware implementation.

Perhaps the most obvious use for volumetric backprojection is the illumination of volumetric data. While local illumination only considers a local neighborhood of information to determine the shading of a point in space, global illumination considers the total distribution of illumination energy throughout space, taking into account both direct and indirect light source visibility from every point. Global illumination, unlike local illumination, maintains a balance between the amount of energy emitted from sources and absorbed by the volume. Not only does this balance lead to more natural illumination, but also to a more convincing and understandable image. Volumetric backprojection is used to transport illumination through the scene, starting from the light sources and propagating outward.

Another use for volumetric backprojection is reconstruction, that is, the process of synthesizing volume data based on global information, usually in the form of projections. A prime example of reconstruction is computed tomography, in which a set of x-ray projections are combined to yield a complete volume dataset. Reconstruction can be based on not only x-ray wavelengths of light, but also on visible wavelengths. For example, a set of standard photographs can be combined into a volumetric model by the process termed voxel coloring [19].

1.1 Prior Work

Backprojection is usually performed on a voxel-by-voxel basis, since this is the most obvious and direct method of computation. For example, in volumetric ray tracing [20] as illumination is computed for a volume sample, rays are cast toward the light sources sampling the partial visibility of each. In computing high-albedo scattering illumination, Max [13] used the method of discrete ordinates to transport energy from voxel to voxel, computed voxel-by-voxel propagated simultaneously in each voxel layer. For calculations of volumetric radiosity, voxels are usually regarded as discrete elements in the usual radiosity calculation on pairs of elements, thereby computing on a voxel-by-voxel basis [18, 21].

Backprojection is also computed using ray-by-ray computations. Recognizing the coherence among the voxels of a volume slice, Cabral et al. [2] performed reconstruction by 2D backprojecting a set of 1D images (a set of rays) using graphics hardware acceleration. This amounts to a ray-by-ray reconstruction, except that a subset of rays is computed simultaneously by the hardware. No methods currently exist that exploit hardware to trace a set of reconstruction rays simultaneously through a *volume*, which could utilize the maximum available coherence. Particle tracing methods for global illumination track paths of scattered light energy through space starting at the light sources [6]. Such ray-by-ray computations, while flexible, are incoherent and can be inefficient.

In many cases, the backprojection can be reorganized into a single sweep through the volume, processing slice-by-slice. Because sunlight travels in parallel rays in one direction only, Kajiya and Von Herzen [7] calculated the light intensity of a cloud-like volume one horizontal slice at a time. A similar technique was demonstrated as part of the Heidelberg ray-tracing model [14] in which shadow rays were propagated simultaneously slice-by-slice and in the same general direction as rendering. Behrens and Ratering [1] implemented efficient slice-by-slice shadowing using texture mapping hardware.

Other volume rendering architectures [10] and hardware [16, 17] are based on slice-by-slice processing, although they perform projection for rendering, rather than backprojection. In particular, Cube-4 [17] utilized a unique skewing scheme which assigned any axis-aligned beam of voxels to a set of parallel distributed pipelines, enabling fully scalable volume processing to occur beam-by-beam and thus slice-by-slice.

In the area of polygon graphics, some have used projection hardware to accelerate global illumination computations. For example, Keller [9] probabilistically created a set of virtual point lights as sources of indirect illumination and rendered a scene in multiple passes. While the same idea could be directly applied to volume rendering, it would be helpful to store view independent illumination information in the volume for subsequent rendering. Szirmay-Kalos [22] used polygon hardware acceleration to accelerate complex radiosity calculations by alternately shooting and gathering illumination based on a quasi-Monte Carlo sequence. Our method similarly builds upon the coherence of ray bundles for the purpose of global illumination, among other things.

1.2 Contribution

In this work, we develop new directions for volumetric backprojection, building on the efficiency of the basic technique. We show how a sequence of simple backprojections can effectively compute a complex volumetric radiosity distribution. We also show how reconstruction techniques can be improved by utilizing volumetric backprojection. We develop a new form of reconstruction which can build a volumetric representation of opaque and translucent colored elements from photographic images. This new technique properly handles the partial volume effect and adapts to diffuse shading.

Given the simplicity and bulk of the computation, backprojection lends itself to a hardware implementation. We suggest some simple modifications to a Cube-4-like volume rendering architecture which would enable volumetric backprojection in addition to standard rendering. These modifications build on the efficiencies of the original design, leveraging them for backprojection.

The following is an overview of the remainder of the paper. In Section 2 we discuss a new algorithm for volumetric global illumination based on a sequence of backprojection iterations. In Section 3 we introduce the fundamentals of computed tomography as a basis for reconstruction from images and develop a new technique for the recovery of scenes from photographic images. In Section 4 we suggest a simple modification to volume rendering hardware to enable all of these applications of volumetric backprojection. Finally, we present our results and discussion in Section 5 and draw our conclusion in Section 6.

2 Illumination by Energy Backprojection

In local illumination, the global distribution of light energy is ignored and shading calculations are performed assuming full visibility of all light sources. While this is useful as a first approximation, the incorporation of global light visibility information (shadows, one instance of global illumination) adds a great deal of intuitive information to the image. This low albedo [7, 20] lighting simulation has the ability to cast soft shadows by volume density objects.

Generous improvements in realism are achieved by incorporating a high albedo lighting simulation [7, 21], which is important in a number of applications (e.g., clouds, skin [5], and stone [4]). While some of these used hierarchical and deterministic methods, most of these simulations used stochastic techniques to transport lighting energy among the elements of the scene.

We wish to solve the illumination transport equation for the general case of global illumination. The incident illumination $I(\gamma, \omega)$ in direction ω at any voxel γ can be described as

$$I(\gamma, \omega) = \int_V \int_\Gamma f(\omega, \omega') I(\gamma, \omega') d\omega' dv$$

where Γ is the set of all directions, V is the set of all voxels v, and $f(\omega, \omega')$ is the phase function in directions ω and ω'. This means that the illumination at any voxel is dependent upon the illumination at every other voxel. In practice, this integral-equation is solved by finite repeated projection of energy among voxels. This leads to a finite energy transport path, which is generally sufficient for visual fidelity.

We make some of the same assumptions as standard radiosity. In our case, we assume that voxels generally behave as diffuse surfaces when a gradient exists. When there is no gradient (as in the case of homogeneous fog) then the voxel scatters light in all directions isotropically.

The approach we take is to organize the computation not per pixel or per voxel but per direction. Organizing per direction allows us to capitalize on coherence by utilizing slice-by-slice computation.

2.1 Direct illumination pass

We begin by first analyzing our volumetric scene to determine the initial distribution of lighting energy. We would like to compute the direct illumination (typically the major contributor to overall intensity) directly. For directional light sources a single sweep similar to [7] along one major axis is sufficient to propagate the light energy to all the voxels. For point light sources both inside and outside the volume, we backproject the light intensity outward from the light source to every voxel using a slice-based approach [10]. However, we have found that in practice it is far simpler to shoot one or more rays toward each of the N^2 exterior voxels of the volume and account for the inverse-square intensity falloff of each ray.

Besides the volume density array $\rho(s), s \in R^3$, we maintain a radiosity array $I_r(s)$ and an unshot radiosity array $I_u(s)$. A transfer function converts each sample volume density ρ_i into an opacity α_i and color C_i. For many datasets, a simple linear ramp from zero opacity at density ρ_a to full opacity at ρ_b is sufficient. For CT datasets, we found it useful to set ρ_a at about 20eliminate noise. For voxelized datasets, the full dynamic range was used. In our tests, we used only a single wavelength of light with objects of a constant intensity. In any case, a transfer function should be chosen for the illumination transport which elucidates the features of interest, the same as in direct volume rendering. As a matter of implementation, the single density value could be replaced with pre-classified RGBα values to support pre-segmented volumes (e.g., the visible human dataset).

In the initial sweep of direct illumination, light energy is transported in proportion to the optical path length to the light source. Borrowing from [7], the radiosity deposited into each voxel along a path from the light source to s is

$$I_r(s) = e^{-\int \kappa(t) dt} \tag{1}$$

where $\kappa(s)$ is the extinction coefficient at s. This is computed incrementally along the path using standard compositing to accumulate opacity along the ray. As energy is depleted from each ray it is deposited into both the radiosity array I_r *and* the unshot radiosity array I_u modulated by the reflectivity λ of the volume sample. The extinction coefficient and reflectivity are both determined by a transfer function based on the local volume density. Note that trilinear or better interpolation should be utilized for both sampling the density ρ and depositing the energy into the radiosity I_r and unshot radiosity I_u arrays.

For area light sources we take a different approach. To compute the direct illumination contribution of an area light source requires integrating across the entire area for each visible voxel. As this is nearly as difficult as calculating the indirect illumination, we postpone the integration until the next step by summing the energy directly into the radiosity and unshot radiosity I_u arrays. If all light sources are area light sources, then we can avoid the initial pass and proceed directly with the indirect passes. However, the smaller our area light sources, the longer it will take to reach equilibrium. Therefore, smaller area light sources can sometimes be more efficiently computed as a small set of point lights.

2.2 Indirect illumination passes

In the second pass we attempt to integrate the illumination contribution of all voxels to all other voxels by a finite number of iterations. In each iteration, we select a random direction σ for our backprojection. Note that the convergence could be improved by selecting directions using a quasi-random (e.g., [8]) sequence of directions rather than a uniform random sequence. An obvious method is to select points distributed on a sphere as directions.

In each iteration, we process slices perpendicular to the major axis nearest to the random direction σ. Starting with the first slice, we initialize a ray front in the form of a 2D buffer. This buffer is used to transport energy along the rays defined by the elements and σ. At each slice, the rays simultaneously accumulate and deposit energy from the neighboring voxels. The differential equation describing the energy E transfer in a ray over a differential length ds is:

$$\frac{dI}{ds} = \begin{cases} -\kappa(s)E(s)\phi(s,\sigma) & \text{if } |\nabla\rho| < 0, \\ I_u(s) - \kappa(s)E(s) & \text{if } |\nabla\rho| = 0, \\ I_u(s)\phi(s,\sigma) & \text{if } |\nabla\rho| > 0, \end{cases}$$

where $\phi(s,\sigma)$ is a function describing the tendency of a volume sample to emit or receive energy in the given direction. Fortunately, this equation is easily solved by finite differences, although it could equally well be solved by a finite element method. The gradient-based energy transfer equation is described next.

In a very high resolution lighting simulation, it would be possible to purely absorb and emit light isotropically by each voxel. This is akin to using microgeometry to determine the reflectance behavior of surfaces. But it is much more efficient to compile statistics on surface reflectances and use a bidirectional reflectance distribution function (BRDF) instead to model the gross effects of the microgeometry. In the absence of surfaces (where there is a zero gradient), we use a simple isotropic absorption-emission model. But at surface boundaries, we allow the energy transfer to only occur in one direction. The ray energy is only allowed to be deposited onto the surface if the ray is approaching the surface. Conversely, unshot radiosity is only allowed to augment the ray energy if the ray is leaving the surface. Additionally, we model surfaces as ideal diffuse reflectors, and therefore we take into account the angle of incidence using the dot product. This distinction between isotropic and diffuse reflectors is automatic, in contrast to Sobierajski's [21] method of explicitly storing two coefficients per voxel.

ζ is used to distribute energy over several iterations. By only emitting part of the voxel radiosity in each iteration, the energy is distributed to a larger variety of voxels, leading to faster convergence. The complete pseudocode algorithm for a single backprojection is given in Algorithm 1. In our implementation, the *ray buffer* contains a slice-sized array of rays which are resampled for interaction with each voxel. Because of the bidirectional transference of energy between the rays and the volume, at least one of the participants must be resampled so that the exchange can take place at a specific location in space. We have chosen to resample the ray buffer because it is 2D, requiring only bilinear interpolation instead of trilinear interpolation of the volume, or both.

```
Procedure Backproject(volume, direction)
  Initialize sheet buffer
  For each slice
    For each voxel in slice
      classify voxel color, opacity, and reflectivity
      determine corresponding ray buffer location
      wrap around using modulo operator
      clear energy of rays that just entered the volume
      If voxelOpacity > 0
        // exchange energy between ray and voxel
        compute dot product of ray direction and gradient
        If dot < 0
          // energy from ray transferred to voxel
          energyRayToVoxel = voxelOpacity×rayEnergy×φ(s,σ)
        Else If dot = 0
          // no surfaces, just isotropic
          // absorption and emission
          energyRayToVoxel = voxelOpacity×rayEnergy
          energyVoxelToRay = voxelUnshot×ζ
        Else If dot > 0
          // energy from voxel transferred to ray
          energyVoxelToRay = voxelUnshot×ζ×φ(s,σ)
        End If
        // store new voxel quantities
        voxelRadiosity += energyRayToVoxel
        voxelUnshot += voxelReflectivity×energyRayToVoxel
        voxelUnshot -= energyVoxelToRay
        // bilinear splat new ray quantities
        rayEnergy += energyVoxelToRay - energyRayToVoxel
      End If
    End Loop
  End Loop
End Procedure
```

Algorithm 1: *Volumetric backprojection algorithm in pseudocode.*

In the procedure, energy exchange is computed one slice at a time, then the ray array is shifted along the ray direction to the next slice as indicated in Figure 1. Parts of the ray buffer which move outside the volume are wrapped around to the other side and re-initialized. A modulo operation efficiently computes the wrap-around.

Clearly, the final distribution of energy will be strongly correlated to the initial chosen direction. If a certain voxel density gradient happens to be in the same direction as the initial direction σ, then all of the unshot energy will be shot in the initial iteration. We use two techniques together to reduce this effect. First, a small value of ζ helps to spread out the contribution over more voxels. Second, we repeat the process many times and average the result. To repeat this without using additional buffers, the total amount of energy added to the system is retained and used to normalize the individual voxel radiosity during rendering. This permits incremental refinement of the solution to include in increasing variety of directional sampling over time.

Because this iterative approach is related to progressive refinement [3], we have the ability to display intermediate results and terminate early if so desired. As in progressive refinement, intermediate stages are visualized by estimating the distribution of energy throughout the scene. Instead of simply splitting the unshot radiosity equally among all the voxels, we wish to avoid placing radiosity in the interior of solid objects. We do this by proportioning the energy according to the product of density and gradient. In this way, empty voxels (which conventionally have zero density) are avoided

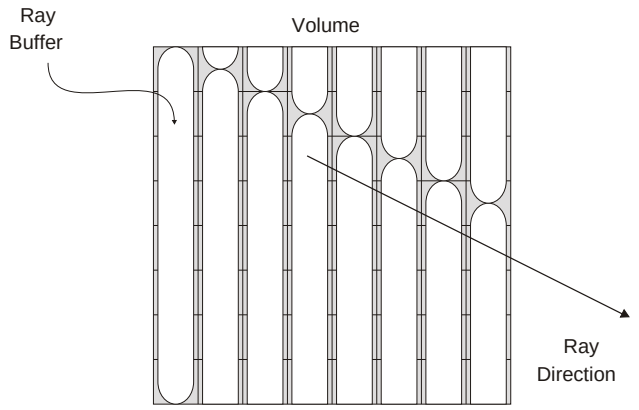

Figure 1: *The ray buffer steps through the volume one slice at a time, wrapping around at the edges.*

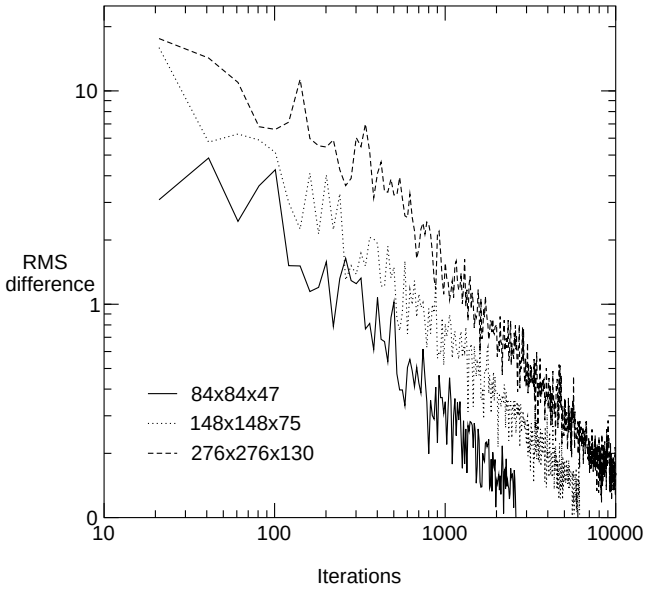

Figure 2: *Convergence of radiosity in the engine scene.*

as well as solid interiors (which usually have no gradient).

The iterations are continued until convergence. Convergence is defined by the voxel-wise root-mean-square (RMS) difference between radiosity estimates Δi iterations apart being below some threshold δ. The RMS difference is computed by the Pythagorean sum of squared differences between corresponding voxels, assuming the original volume is in the range [0,1]. Of course, termination can be accelerated by accepting a greater error tolerance and vice versa, leading to an adjustable time-quality tradeoff.

Selecting $\Delta i \geq 20$ is used to avoid local minima in the search process. Figure 2 demonstrates the logarithmic rate of convergence with t=20 and δ=0.1. When convergence is achieved, there is usually unshot radiosity in the scene from the last several iterations; the radiosity added in each iteration has a half-life which is data dependent. The unshot radiosity can be (1) ignored and removed from the sum of unshot radiosities, (2) distributed among the other voxels of the scene, or (3) distributed more appropriately by iterating further until some proportion ϵ of the total energy is dissipated. The latter is the most appropriate technique, but this choice has little effect on the final distribution after convergence.

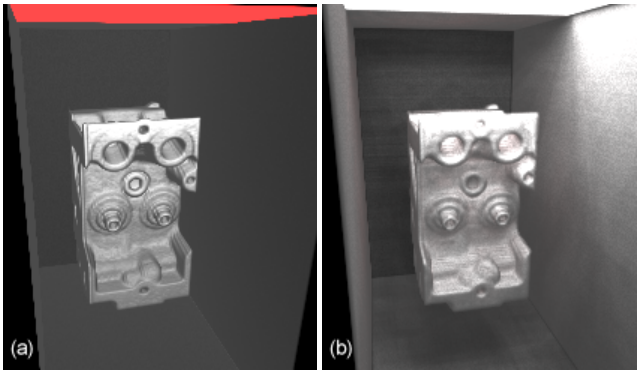

Figure 3: *(a) Initial configuration of the engine block scene before any lighting simulation, and (b) after 1000 iterations. (See also in the color plates.)*

2.3 Rendering

To render using the radiosity-density representation, we use a modified version of direct volume rendering [12]. Instead of shading each sample along the ray by summing the illumination by each of the light sources, we just use the pre-computed radiosity which already contains the influence of all the light sources, both direct and indirect. The image rendering equation from point s_0 in direction σ is then:

$$I(s_0, \sigma) = \int_{s_0}^{\infty} \rho(s) I_r(s) e^{-\int_{s_0}^{s} \rho(t) dt} ds$$

We found that we could enhance the image contrast, emphasize the gradient, and improve the overall appearance by including a $\cos(\theta)$ factor in the integral, similar to Lambert's law. θ is the angle between the viewing ray and the volume gradient. It is computed using the dot product $\nabla\rho(s) \cdot \sigma$ clamped to the range [0,1]. In the absence of a volume gradient a value of 1 is used in place of the dot product, for this indicates a homogenous region that emits isotropically.

Figure 3a shows the initial configuration of the test scene including the CT scanned engine block in a translucent box. An area light source was modeled as half the ceiling. Figure 3b demonstrates rendering the test scene after 1000 iterations. Note the soft shadows on the object and the indirect illumination of the interior of the object. Notice that some radiosity (in red) continues to bounce in the interior of the object and crevices. This small amount of the total energy (0.02%) was amplified for visualization and can safely be ignored without consequence.

3 Reconstruction by Image Backprojection

A number of methods have been proposed to reconstruct the 3D shape of objects from photographic images. Kutulakos and Seitz [11] use a technique called space carving to generate a binary representation of the objects on a discrete volume grid. It works by backprojecting the object's silhouette edges that can be detected in the images. Seitz [19] proposed a method termed voxel coloring that works its way through the scene from front to back in layers and picks for each voxel its most likely color, given the acquired images. Both methods make a binary decision on what color and occupancy a voxel should have, which can lead to aliasing. In this section, we would like to explore new approaches to reconstruct a volumetric object from its projections.

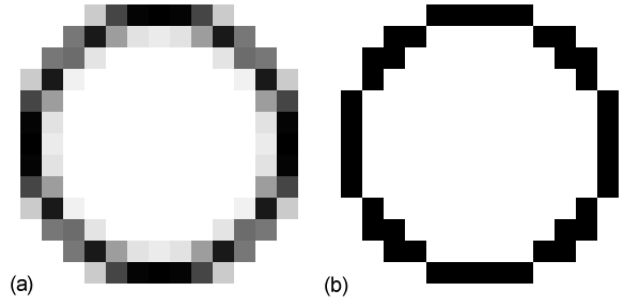

Figure 4: *A section from (a) continuous and (b) binary object reconstructions.*

In making binary decisions about the color and occupancy of a reconstructed voxel, we discard important clues. The degree of certainty about a voxel is better encoded into the opacity. We treat the voxels as point samples of a continuous field. Only interior voxels should be labeled as fully occupied; voxels on the surface should be partially occupied and indicated by partial opacity. The final voxel opacity should be a weighted average of the estimations given by the projections. Due to the low-pass filtering inherent in image acquisition, all reconstructed objects will exhibit antialiasing. For example, Figure 4 shows a section of reconstructed voxels from a hoop using continuous and binary occupancy decisions.

We start by observing that reconstruction is a common procedure in the medical field. There, computed tomography (CT) is routinely employed to recover a patient's interior from X-ray projections that were taken around a circular orbit around the patient. The most commonly used CT method is Filtered Backprojection (FBP), where the projections are first filtered with a high-pass filter, and then backprojected onto the volume. The high-pass filtering is necessary to avoid blurring of the reconstructed object, and the backprojection can be thought of as a simple spreading of the filtered projection image across the volume grid. The theory behind FBP requires the projection images to be spaced at equidistant orientations around the patient. The quality of the reconstruction suffers considerably when this prerequisite is not fulfilled, and also when the number of projections is small (that is why 500 and more projections are taken by medical scanners). In these scenarios, iterative techniques, such as the Simultaneous Algebraic Reconstruction Technique (SART), are more adequate. In SART, the volume is reconstructed by a sequence of projections and backprojections. The technique iteratively (1) projects an image from the volume currently being reconstructed, (2) compares it to the actual X-ray image acquired from the scanner, (3) corrects the reconstructed volume using backprojection, and (4) repeats the process until convergence.

To implement SART, a sequence of x-ray images is selected; convergence is faster if successive images are projected in approximately orthogonal directions. A relaxation factor $\lambda \in [0, 1]$ is selected to mix each voxel with its correction. For each image in the sequence, the existing volume (initially empty) is projected from the same viewpoint as the x-ray image. The true image is subtracted from the approximate image and the result scaled by λ. This difference image corresponds to the correction which would fix the volume according to that viewpoint. Rays traverse the volume and deposit the correction value (either positive or negative) to the voxels along the ray. As the process is repeated, the volume converges to the original sampled volume.

CT can reconstruct three-dimensional object features of very little contrast (less than 0.5%) and with high resolution (less than 1mm), but tomographic reconstruction is primarily used in the context of imaging with X-ray energies which are confined to hospitals and shielded industrial sites. Apart from the fact that X-rays are difficult to generate, health considerations prohibit us to use X-ray technology to scan real objects in the office, home, or laboratory, for subsequent incorporation on our graphics scenes. The question is, can we use the high-fidelity properties of CT methods to reconstruct objects imaged with harmless visible light and so recover low-contrast and very detailed object features.

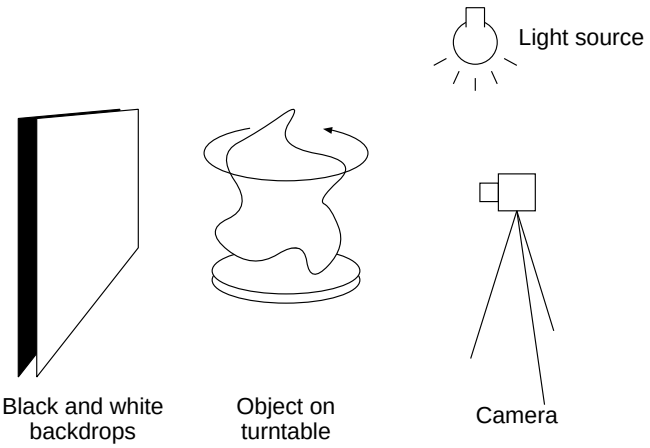

Figure 5: *Setup of the opaque reconstruction virtual test rig.*

Since all CT methods including SART assumes all objects can be perfectly penetrated by the X-ray beam, obscuration is not a problem. But, using visible wavelengths of light means that some parts of the scene may be obscured in some or all of the images. For that reason, the estimated volume usually never approaches the real volume because the interior is indeterminate. The same problem arises with reconstruction from saturated x-ray images. Furthermore, some parts of the scene may be indeterminate due to specular highlights (e.g., a mirror) or complete shadowing.

We adopted a virtual test setup as shown in Figure 5. A scene of random translucent triangles are voxelized into a reference volume. Then, a virtual light source, camera, and backdrop are positioned in the scene. The volume is rotated on a virtual turntable to acquire a non-uniform sequence of projections with both a white and a black backdrop and controllable ambient and diffuse shading. A reconstruction volume containing both color and opacity is initialized to empty. Then a number of iterations are used to converge the reconstruction volume.

In each iteration, a random source projection is selected and virtually imaged. Although we can obtain the opacity with volume rendering, it is unavailable with standard image capture. Using two images, one with a white backdrop and one with black, we can compute the opacity afterward with a straightforward derivation involving the compositing operator. Given a pixel of a photograph of the object over a white background C_w and over a black background C_b, we express them in terms of the object color C_o, the object opacity α_o, and the compositing equations

$$C_w = C_o\alpha_o + 1(1 - \alpha)$$
$$C_b = C_o\alpha_o + 0(1 - \alpha)$$

and solving for α we get

$$C_o = \frac{C_b}{\alpha_o}$$
$$C_w = \frac{C_b}{\alpha_o}\alpha_o + 1 - \alpha_o$$
$$\alpha_o = C_b + 1 - C_w$$

A corresponding projection is made from the reconstructed volume assuming some ambient and diffuse shading coefficients. The source opacity and color are compared to the reconstructed opacity and color and correction values are generated for each pixel, modulated by λ as in SART. The correction opacity and color are backprojected through the volume and applied to the voxels along each ray. All processing was performed using a simple and efficient slice-based technique.

The similarities between our global illumination technique (see Section 2) and iterative reconstruction are strong: Both (1) require a set of directionally varying backprojection operations, (2) iterate until convergence, and (3) can benefit from an efficient slice-based processing scheme. The differential equation describing the energy transfer in a reconstruction ray over a differential length ds is just a constant function $\Phi(u, v, \omega)$ of pixel position (u, v) and direction ω which is computed per once per ray per iteration. Φ is computed as the difference between the pixel (u, v) from the real scanner in direction ω and the integral of the corresponding ray in the estimated (reconstructed) volume. An entire projection or backprojection can be performed efficiently by sweeping through the slices with a rayfront as described above.

4 Hardware

Clearly, the efficient parallel volume processing capabilities of a Cube-4-like design could be utilized to improve the performance of volumetric backprojection. Cube-4 enabled linearly scalable volume rendering by the addition of similar distributed memory and processing elements. We will now briefly review the pertinent architectural points.

Cube-4 processes a rayfront of rays one slice at a time (internally it processes one beam at a time). The hardware utilizes coherence to efficiently resample volume data for the rays. As each beam of voxels is read by a beam of processors, they share volume data with one neighbor and immediately interpolate new samples. A beam buffer retains a copy of the samples so that in the next cycle when the next beam is read, a new beam of samples can be interpolated to align with the rays. These new samples are again buffered in a slice buffer until the corresponding samples are available in the next slice. Finally, samples are interpolated in between the slices. Gradients are efficiently computed in a similar fashion. Finally, samples are accumulated into a compositing buffer which contains the ray colors and opacities.

Backprojection can be efficiently performed within this framework. Essentially, the pipeline is run in reverse. The rays are initially loaded into the compositing buffer. In each cycle a beam of rays are read from the compositing buffer and resampled to match the voxel grid. 2D resampling is available already in the compositing buffer using two beam buffers and ray sharing among processors. At this point, the rays are aligned with the voxels and compositing can occur. If reconstructing, then the ray energy is simply added to the voxel; if illuminating with visible light, the ray energy is composited based on the voxel density. In either case, the voxel value must be read, modified, and written back. This requires twice the voxel bandwidth of plain rendering, so it proceeds at half the rate.

In illumination with visible light, the ray energy is dissipated along the ray based on the local volume density, so the ray energy must also be modified and written back before the next slice. This can be done using the voxel resampling hardware which is so far unused. As voxels are read, they are buffered and 2D interpolated to match the ray grid (at the same time that the ray grid is 2D interpolated to match the voxel grid). When resampled voxel match the ray grid, they diminish the intensity of the corresponding ray. Again, this requires twice the buffer bandwidth to read, modify, and write back the ray energies, so it can only proceed at half the rate of rendering, but the speed is already limited by the voxel bandwidth. With these techniques we can design hardware to perform efficient, scalable reconstruction and global illumination.

5 Results and Discussion

5.1 Global Illumination

The core of the global illumination backprojection algorithm was implemented in unoptimized C++ code. A variety of tests were run on various datasets to determine appropriate parameters. For instance a useful value of ζ was found to be 0.1. It was also found that no more than 1000 iterations generally suffice for visual fidelity.

The final radiosity distribution is the average of the energies distributed in each iteration. The separability of the average operation indicates that parallelism is possible and efficient. Assuming the whole dataset fits into the memory of each processor, interactions can easily be distributed to multiple processors and combined in the end with near linear speedup. However, convergence is more difficult to control since serialization is necessary to establish the quality of the distribution.

To speed convergence, it is helpful to select directions which would transport the most energy. But, to fairly represent all directions, importance sampling should be used. Importance sampling and parallel computation are relegated to future work.

For most of the computations except the final rendering, zero-order interpolations were used. This dramatically cuts down on the computation time with little degradation in accuracy. A state-of-the-art hardware volume rendering accelerator could be modified to support backprojection at about half the usual frame rate. Although computing radiosity for the full size engine volume required 2.3 h in software, a hardware accelerator could likely complete the task in under a minute, making it relatively "instant" [9].

The beauty of the volumetric approach to global illumination is that is provides us with a regular computational space, relatively insensitive to the actual data. Furthermore, we are free to regulate the resolution of the computation based on the desired artifacts. That means to generate primarily low frequency artifacts, such as the distribution of indirect illumination in a scene, we can choose to compute at a lower resolution and simply scale the results back to the final resolution for final rendering. Such a feat would be very difficult with surface-based graphics.

The price of reduced resolution is a loss of high frequency detail. However, radiosity is a relativelyl low frequency phenomenon. As long as the selected computation resolution is sufficient to capture the desired phenomenon, there is no visible loss of accuracy. For example, computing on a $276\times276\times130$ volume took an average of 8.4 s per pass, while a reduced resolution $148\times148\times75$ volume took 1.5 s per pass. Figure 6 compares the artifacts generated by computing on the low and high resolution datasets. The final rendering used the high resolution dataset for density and the low resolution for radiosity information. The primary visual difference is the inability of illumination to reach into the smallest crevices. However, the net effect is virtually indential.

An important question to answer is whether improved illumination methods result in improved visualizations. This subjective question is best answered by way of demonstration. Figure 7 demonstrates the difference in visualization between using only local and global illumination. Note how the global illumination improves spatial perception. Although Figure 7a casts soft shadows, the complete darkness of shadows obscures objects in recesses. In Figure 7b the recesses are properly illuminated through diffuse inter-reflection giving an intuitive understanding of depth and accessibility. Other details include the indirect illumination of the ceiling and the illumination streaming through the central hole in Figure 7b.

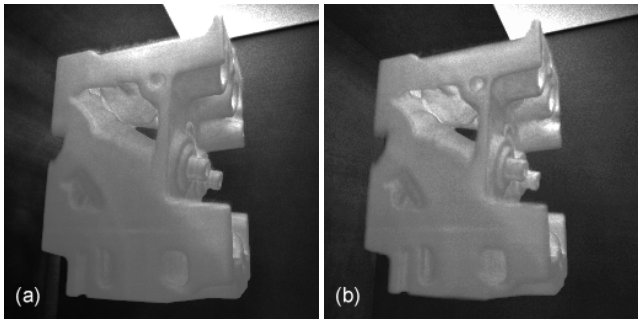

Figure 6: *Comparison of the radiosity artifacts between (a) low and (b) high resolution datasets. (See also in the color plates.)*

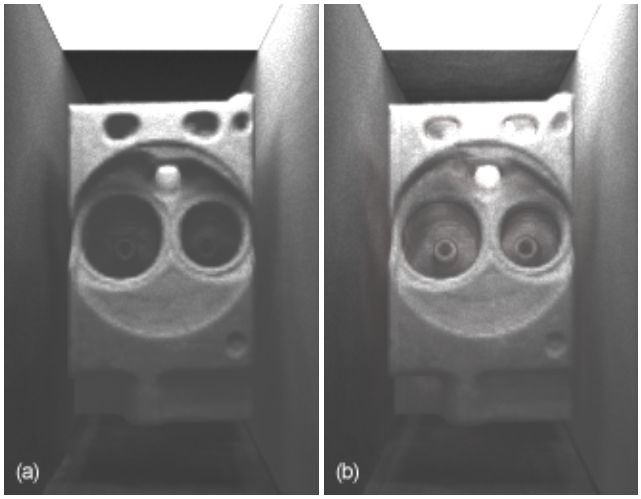

Figure 7: *Comparison of (a) single scattering, and (b) multiple scattering from an area light source. (See also in the color plates.)*

The global illumination algorithm attempts to solve a radiosity problem similar to Rushmier and Torrance's zonal radiosity [18]. In our case, we are only dealing with volumetric data, thus we can employ efficient slice-based computation. Sobierajski's [21] hierarchical radiosity scheme supports both surfaces and volumes and hierarchical methods in a deterministic computation step. We rely on stochastic methods to transport illumination throughout the scene, whereas [18, 21] compute deterministically. Because our algorithm only supports a volumetric primitive, it can be far simpler to implement than others.

However, we do support voxelized surfaces (the walls are voxelized from surface representations). By first voxelizing it becomes easier to control complexity by using alternate volumetric resolutions. Radiosity computation can occur at a low resolution, but the high resolution volume data and exact surfaces can be used for final rendering. The radiosity data can be used as a 3D texture for the surfaces.

The relative efficiency of our scheme is highly data dependent. Because zonal and hierarchical methods rely on an adaptive discretization of the scene, highly complex scenes may generate a large number of discrete elements and data structures. Conversely, we maintain only the original dataset in its native (or reduced resolution form) and compute solely on that. The only other data structures required are to store the volumetric radiosity and unshot radiosity data. Thus, we achieve a clean and simple implementation free from complex data structures. However, datasets which are homogenous or largely empty represent an inefficiency when computed at full resolution.

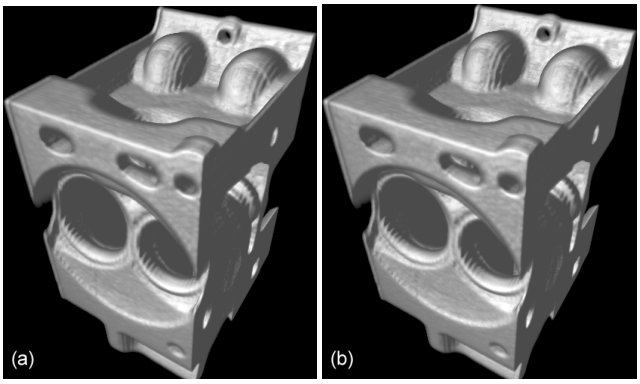

Figure 8: *(a) Original volume dataset, and (b) reconstruction after 1000 iterations. (See also in the color plates.)*

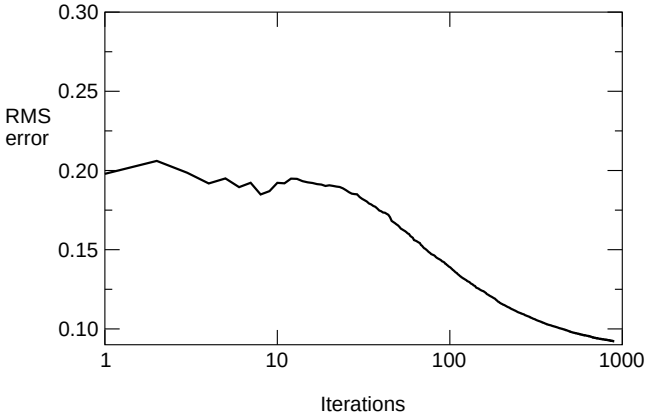

Figure 9: *Convergence of SART reconstruction of a 256×256×110 engine block dataset.*

5.2 Reconstruction from Images

We tested the SART reconstruction technique first with synthetic x-ray projections on a 256×256×110 engine block dataset (see Figure 8). After 1000 backprojections which averaged 11.2 s each we achieved an RMS error of 0.09 (see Figure 9). If the same 1000 iterations were performed using backprojection hardware as described in Section 4, it would complete in just over one minute.

Numerical precision is necessary for accurate reconstruction. Medical reconstructions are worthless unless they can provide very high contrast ratios. For that reason, many bits of fixed precision are required to store each voxel. Tests [15] have shown that 16 bits of scalar data are required to provide a 1% percent contrast sensitivity in reconstruction. With this much precision in the hardware, hardware accelerated medical reconstruction is practical.

To be prepared for the future, hardware should be able to perform cone beam reconstruction which requires perspective projection. The fundamentals of backprojection remain the same; energy is deposited along rays. Only the rays diverge and care must be taken to ensure that no voxels are missed [10].

Next, we simulated 1000 iterations of visible light reconstruction with 150 randomly placed, randomly colored and randomly translucent triangles in a 128^3 volume to form a mobile (see Figure 10a). By virtually imaging the scene we were able to concentrate on the novel reconstruction algorithm rather than the calibration and noise

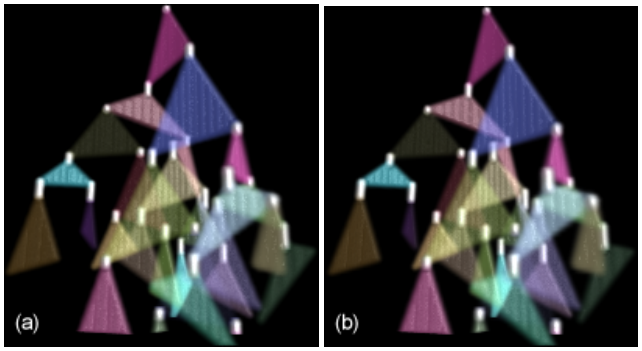

Figure 10: *(a) Source volume dataset of 100 randomly placed, randomly colored, and randomly translucent triangles and (b) reconstructed volume after 1000 iterations including proper translucency. (See also in the color plates.)*

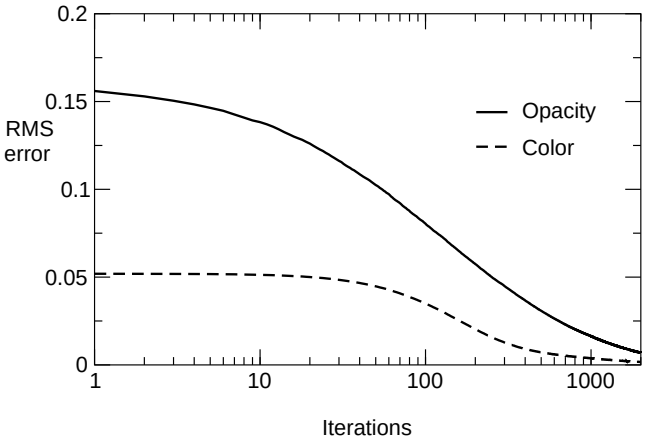

Figure 11: *Error in visible light reconstruction during convergence.*

supression required by real world imaging. Orthographic projections were acquired from random directions in a single plane. Using additional degrees of freedom increases the potential quality of reconstruction, but limits the practical utility of the technique. Figure 10b demonstrates the result of reconstruction of the random triangle scene. Figure 11 shows how the RMS error decreases over the iterations. The error in the color was based on the opacity weighted color (premultiplied α), since that more closely relates to the visual impression. For the color to properly accumulate requires some amount of opacity to be present, therefore, the convergence of the color lags behind the opacity convergence.

Of concern is the fact that diffuse and specular shading changes the perceived color depending upon orientation and this would influence the reconstruction process. Specular reflections are a complicated phenomenon more appropriate for computer vision research, so we restricted ourselves to diffuse materials only. We varied the ratio of diffuse to ambient shading from 0 to 1, performed a fixed number of passes, and compared the results of reconstruction of the triangle dataset. There was no appreciable influence on the quality of reconstruction (see Figure 12).

Because the rendering of the estimated reconstruction volume is performed with diffuse shading enabled, the algorithm was able to factor out the effect of shading. In effect, shape is somewhat derived from shading. Without precisely knowing the light source reconstruction can be tricky. However, this can be another dimension to the reconstruction process. While it would take much longer

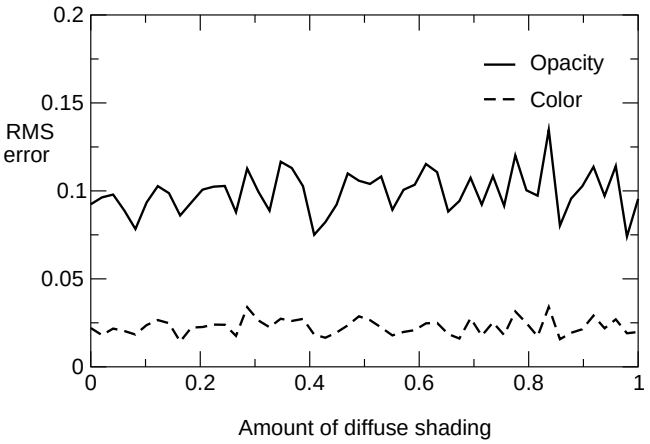

Figure 12: *The negligible effect of diffuse shading on reconstruction quality.*

(essentially repeating the entire process for each possible light position), it is possible.

In contrast to other visible light reconstruction techniques, our method properly includes the partial volume effect. That is, the soft edges obtained by a real-world sampling process are reflected in the scalar opacity volume buffer. While other methods only distinguished between occupied and non-occupied voxels, ours recovers soft edges as well as fully opaque voxels. The beneficial result is antialiasing during re-rendering from novel views. As such, re-projections of the volume buffer can be used to introduce soft-edged, foggy, and translucent materials as a supplement image-based rendering methods.

6 Conclusion

Volumetric backprojection has been used in the past for efficient shadow computations. We have shown how this technique can be used to solve a variety of radiative transport and even reconstruction problems. Specifically, we have shown how to simply and effectively compute global illumination for complex datasets. We have taken advantage of lower resolution computation to improve the performance without appreciable penalty in results. We have implemented the technique of SART CT reconstruction using efficient volumetric backprojection. We have developed a new form of reconstruction which operates in the visible light spectrum and permits acquisition of color-opacity volume data. Finally, we have suggested how volumetric backprojection could be designed into a Cube-4-like volume rendering hardware accelerator. A hardware implementation would permit all of these applications and more to operate at unprecedented rates. The integration of backprojection operations with a volume visualization accelerator would allow myriad interactive applications.

Acknowledgments

This work was funded by NSF grant MIP9527694 and ONR grant N000149710402. Klaus Mueller was funded by a SUNY seed grant. The engine block dataset is courtesy of GE. The authors wish to thank Justine Dachille and Bin Zhang for their assistance during the work.

References

[1] U. Behrens and R. Ratering. Adding shadows to a texture-based volume renderer. In *1998 Symposium on Volume Visualization*, pages 39–46. IEEE, ACM SIGGRAPH, 1998.

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

[3] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In J. Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, Aug. 1988.

[4] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. K. Pedersen. Modeling and rendering of weathered stone. *Proceedings of SIGGRAPH 1999*, pages 225–234, Aug. 1999.

[5] P. Hanrahan and W. Krueger. Reflection from layered surfaces due to subsurface scattering. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 165–174, Aug. 1993.

[6] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 311–320, July 1998.

[7] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 165–174, July 1984.

[8] A. Keller. Quasi-monte carlo radiosity. In X. Pueyo and P. Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 101–110. Springer Wein, June 1996.

[9] A. Keller. Instant radiosity. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 49–56, Aug. 1997.

[10] K. Kreeger, I. Bitter, F. Dachille, B. Chen, and A. Kaufman. Adaptive perspective ray casting. In *IEEE Symposium on Volume Visualization*, pages 55–62. IEEE, ACM SIGGRAPH, 1998.

[11] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. Technical Report 692, Computer Science Dept., University of Rochester, Rochester, New York, May 1998.

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

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

[14] H.-P. Meinzer, K. Meetz, D. Scheppelmann, U. Engelmann, and H. J. Baur. The Heidelberg ray tracing model. *IEEE Computer Graphics and Applications*, 11(6):34–43, Nov. 1991.

[15] K. Mueller and R. Yagel. On the use of graphics hardware to accelerate algebraic reconstruction methods. In *Proceedings of SPIE Medical Imaging Conference 1999*, number 3659-62, San Diego, CA, Feb.

[16] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. *Proceedings of SIGGRAPH 1999*, pages 251–260, Aug. 1999.

[17] H. Pfister and A. E. Kaufman. Cube-4 - A scalable architecture for real-time volume rendering. In *1996 Volume Visualization Symposium*, pages 47–54. IEEE, Oct. 1996.

[18] H. E. Rushmeier and K. E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 293–302, July 1987.

[19] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 25(3), November 1999.

[20] L. Sobierajski and A. Kaufman. Volumetric ray tracing. In *1994 Symposium on Volume Visualization*, pages 11–18. ACM SIGGRAPH, Oct. 1994.

[21] L. M. Sobierajski. *Global Illumination Models for Volume Rendering*. Ph.D. thesis, Stony Brook, NY, Aug. 1994.

[22] L. Szirmay-Kalos and W. Purgathofer. Global ray-bundle tracing with hardware acceleration. In *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 247–258, 1998.

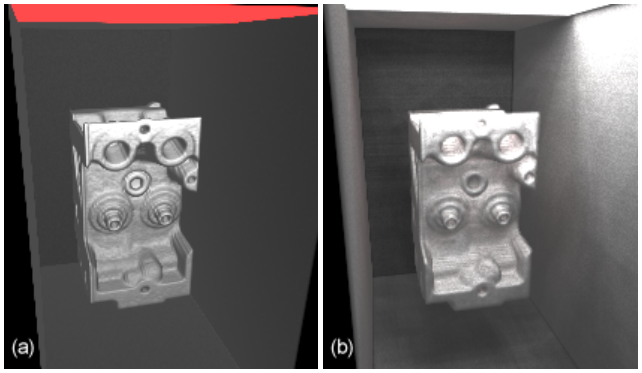

Figure 3: *(a) Initial configuration of the engine block scene before any lighting simulation, and (b) after 1000 iterations. Red indicates unshot radiosity.*

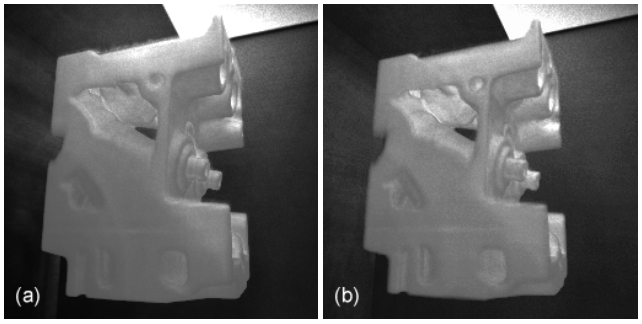

Figure 6: *Comparison of the radiosity artifacts between (a) low and (b) high resolution datasets.*

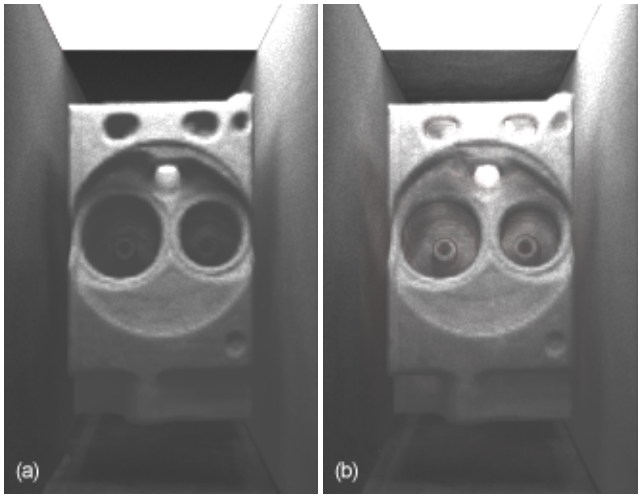

Figure 7: *Comparison of (a) single scattering, and (b) multiple scattering from an area light source.*

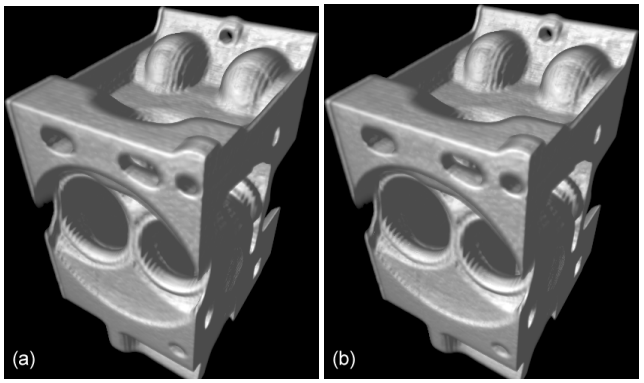

Figure 9: *(a) Original volume dataset, and (b) reconstruction after 1000 iterations.*

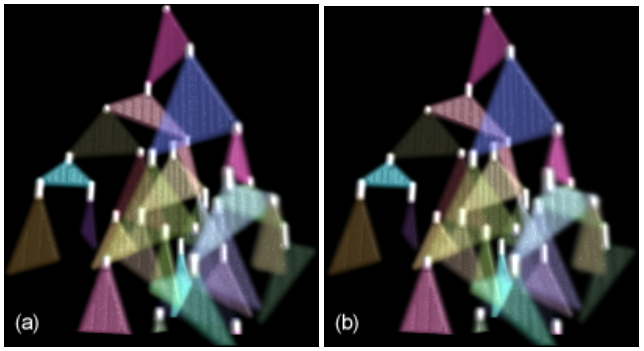

Figure 10: *(a) Source volume dataset of 100 randomly placed, randomly colored, and randomly translucent triangles and (b) reconstructed volume after 1000 iterations including proper translucency.*