Parallel Iteration to the Radiative Transport in Inhomogeneous Media with Bootstrapping

László Szirmay-Kalos, Gábor Liktor, Tamás Umenhoffer, Balázs Tóth, Shree Kumar, Glenn Lupton

Abstract—This paper presents a fast parallel method to solve the radiative transport equation in inhomogeneous participating media. We apply a novel approximation scheme to find a good initial guess for both the direct and the scattered components. Then, the initial approximation is used to bootstrap an iterative multiple scattering solver, i.e. we let the iteration concentrate just on the residual problem. This kind of bootstrapping makes the volumetric source approximation more uniform, thus it helps to reduce the discretization artifacts and improves the efficiency of the parallel implementation. The iterative refinement is executed on a face centered cubic grid. The implementation is based on CUDA and runs on the GPU. For large volumes that do not fit into the GPU memory, we also consider the implementation on a GPU cluster, where the volume is decomposed to blocks according to the available GPU nodes. We show how the communication bottleneck can be avoided in the cluster implementation by not exchanging the boundary conditions in every iteration step. In addition to light photons, we also discuss the generalization of the method to γ-photons that are relevant in medical simulation.

Index Terms—Radiative transport equation, multiple scattering, diffusion approximation, FCC grid, parallel computation, Monte Carlo method, iteration, GPU, CUDA.

1 Introduction

The multiple-scattering simulation in participating media is one of the most challenging problems in computer graphics, radiotherapy treatment design, and in PET/SPECT reconstruction where the scattered component distorts the reconstruction results, thus it needs to be estimated and subtracted from measured data. In these applications the simulation should be fast enough to allow the examination of many source positions in the available time.

Cerezo et al. [3] classified solution algorithms as analytic, stochastic, and iterative.

Analytic techniques rely on simplifying assumptions, such that the volume is homogeneous, and usually consider only the single scattering case [4], [5]. Stam [9] introduced the *diffusion theory* to compute energy transport in optically dense media. This method expresses the radiance by the first two terms of the spherical harmonic expansion, which can be obtained by the solution of a diffusion equation. The diffusion equation can be solved analytically for homogeneous materials and for isotropic point or directional sources, but requires iterative methods in the general case [9], [12]. Based on the analytic solution for the homogeneous case, Jensen et al. [10] attacked the subsurface light transport by assuming that the space is partitioned into two half spaces with homogeneous materials and developed the *dipole model*. Tong et al. [11] investigated the *quasi-homogeneous* problem where the material is homogeneous on the large scale but has high frequency local variations (e.g. a slice of bread). Haber et al. [13] exploited the divergence theorem to replace the volume integrals of each voxel by the surface integral along the voxel faces, which allows better treatment of boundary cells. Wang et al. [14] also attacked the problem of boundary representation in subsurface scattering and distorted the Cartesian grid to a *polygrid* to get boundary nodes to lie exactly on the surface while maintaining the original 6-connection topology.

Stochastic methods rely on Monte Carlo quadrature. In order to get the radiance of a point, all photon paths connecting the source to this point via arbitrary number of scattering events should be considered and their contributions be added. As the location of a single scattering can be specified by three Cartesian coordinates, the contribution of paths of length l can be expressed as a $3l$-dimensional integral. As l can be arbitrary, we should deal with high-dimensional (in fact infinite-dimensional) integrals. Such high-dimensional integrals can be evaluated by Monte Carlo quadrature that samples the high-dimensional domain randomly and approximates the integral as the average of the contribution of these random paths, divided by the probability of the samples. The error of Monte Carlo quadrature taking n samples is in $O(n^{-1/2})$. To speed up the computation by a linear factor, ray samples are reused for many pixels, storing partial results, for example, in photon maps [15], [16].

Iteration obtains the solution as the limiting value of an iteration sequence. In order to store temporary radiance estimates, a finite element representation should be

- *L. Szirmay-Kalos, G. Liktor, T. Umenhoffer, and B. Tóth are with the Department of Control Engineering and Information Technology, Budapest University of Technology, and Economics, Hungary. E-mail: see http://www.iit.bme.hu*
- *S. Kumar is at Hewlett-Packard, India. E-mail: shree.kumar@hp.com*
- *G. Lupton is at Hewlett-Packard, USA. E-mail: glenn.lupton@hp.com*

used. The spatial domain is discretized by a voxel grid, radial basis functions [17], or by particles [18]. Popular schemes for the directional discretization include partitioning the directional sphere and spherical harmonics [19]. The error between the actual and the limiting values reduces with the speed of a geometric series, i.e. it is in $O(\lambda^n)$ after n iteration steps where λ is the ratio of energy decrease in a single radiation–medium interaction. The *zonal method* [20] computes interaction between all spatial finite elements at each iteration cycle, which has prohibitive computational complexity. The *discrete ordinates method* [21], on the other hand, considers just the interactions of spatially close elements in discrete directions. Restricting a single iteration step to local interactions makes the transport matrix sparse and reduces the complexity considerably. Unfortunately, the discrete ordinates method suffers from two types of discretization artifacts unless the number of discrete directions is high. Due to the repetition of the directional interpolation in each iteration step, it *smears* sharp light beams. On the other hand, *spurious beams* or the *ray effect* show up starting at high intensity regions, which reveal the underlying discretization scheme. Note that both artifacts are strong if the source distribution is strongly non-uniform.

This paper proposes an iterative solution to interactively render inhomogeneous participating media defined by large voxel arrays. Iteration has better convergence rate than Monte Carlo particle tracing thus it is more appropriate in speed critical systems. However, its parallel execution is more complicated and it needs some finite element representation to store temporary data, which introduces discretization artifacts. In order to attack these problems, we use a simple and fast technique to initially distribute the radiation in the medium. The distribution is governed by the diffusion theory, where the single pass approximate solution is made possible by assumptions that the medium is locally homogeneous and spherically symmetric. As we use the solution of the diffusion equation only to bootstrap the iteration, our diffusion solver provides just a rough approximation but can be obtained in parallel to the direct term computation at negligible additional cost. Having obtained the initial approximation, the residual of the solution is computed by iteration on a GPU cluster.

Summarizing, the main contributions of this paper are as follows:

- The bootstrapping algorithm that improves the convergence of the following iteration phase, reduces the iteration's discretization artifacts, and makes the parallel implementation more efficient.
- The parallel iteration algorithm that runs on a Face Centered Cubic (FCC) grid and can be considered as a special and fast version of the discrete ordinates.
- The proposition of exchanging boundary conditions not in every iteration cycle in order to eliminate the communication bottleneck in GPU clusters.

This paper is an extended version of [1] which introduced the basic idea of taking a rough solution as the initial value of the iteration. Here we provide the details of the algorithm, improve the robustness of the initial estimation, show that the initial distribution not only helps the parallel execution but also reduces discretization problems, and extend the iteration algorithm to γ-photons that have higher energy levels than light photons.

The paper is organized as follows. Section 2 discusses the theory of radiative transport both for light and γ-photons. Section 3 introduces the new method, explaining the computation of the initial estimation of the radiance and developing the iterative refinement. Section 4 presents our distributed implementation. Section 5 discusses the results, and finally we close the paper with conclusions in Section 6.

2 Radiative transport

This section reviews the physical theory of photon–media interaction. The used notations are also summarized in Table 1.

Symbol	Interpretation
$L(\vec{x},\vec{\omega})$	Radiance of point $\vec{x}$ in direction $\vec{\omega}$
$L_d(\vec{x},\vec{\omega})$	Direct term, unscattered radiation
$L_m(\vec{x},\vec{\omega})$	Media term, scattered radiation
σ_t	Extinction coefficient
σ_s	Scattering coefficient
a	Albedo
g	Extent of anisotropy
σ_t'	Reduced extinction coefficient, $\sigma_t' = \sigma_t - \sigma_s g$
σ_e	Effective transport coefficient, $\sigma_e = \sqrt{3\sigma_a\sigma_t'}$
$\vec{\omega}'$	Incident direction
θ	Scattering angle, $\cos\theta = \vec{\omega}' \cdot \vec{\omega}$
$P(\cos\theta)$	Phase function
$Q(\vec{x},\vec{\omega})$	Volumetric source
h	Planck's constant, $4.14 \cdot 10^{-15}$ eV·s
ν	Frequency of the radiation
E_0	Incident photon's energy, $E_0 = h\nu$
E_1	Scattered photon's energy
$C(\vec{x})$	Electron density at point $\vec{x}$
$m_e c^2$	Electron's energy, $m_e c^2 = 511$ keV
r_e	Classical electron's radius, $2.82 \cdot 10^{-15}$ m
Φ_0	Radiant intensity of the source
$\phi(r)$	Fluence at distance r from the source
$\vec{E}(r)$	Vector irradiance
α	Opacity of the voxel

TABLE 1
Notations used in this paper.

When photons interact with participating media, they scatter either on electrons or less probably on atomic cores. A photon has zero rest mass, but can be associated with relativistic mass $m = E/c^2 = h\nu/c^2$ where E is the energy of the photon, h is the Planck constant, ν is the frequency of the radiation, and c is the speed of light. In case of the visible spectrum, the relativistic mass of a photon is negligible with respect to the mass of electrons or atomic cores, so when a photon elastically scatters on an electron, it bounces off like hitting a rigid wall, keeping its energy and consequently its original

frequency. In case of inelastic scattering, also called the *photoelectric effect*, the photon's energy is absorbed. Thus, upon scattering, the number of light photons reduces but the frequency of the remaining photons does not change. This is why we can handle frequencies independently in computer graphics.

On higher energy or frequency ranges, however, photon energy and impulse become comparable to the energy and impulse of electrons. Thus, scattering may modify not only the number of photons but also their frequency, so frequencies become coupled and cannot be handled independently. This frequency range is particularly important in medical simulation since CT, PET, SPECT, etc. devices work with γ-photons.

2.1 Light photons

For light photons, we can solve the transport problem independently on each of the representative frequencies, usually corresponding to red, green, and blue light. Multiple scattering simulation should solve the *radiative transport equation* that expresses the change of radiance $L(\vec{x},\vec{\omega})$ at point $\vec{x}$ and in direction $\vec{\omega}$:

$$\vec{\omega}\cdot\vec{\nabla}L = \left.\frac{\mathrm{d}L(\vec{x}+\vec{\omega}s,\vec{\omega})}{\mathrm{d}s}\right|_{s=0} =$$

$$-\sigma_t(\vec{x})L(\vec{x},\vec{\omega}) + \sigma_s(\vec{x})\int_\Omega L(\vec{x},\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega'. \quad (1)$$

In this equation the negative term represents *absorption* and *out-scattering* (Fig. 1). The probability of collision in a unit distance is defined by *extinction coefficient* σ_t, which is broken down to *scattering coefficient* σ_s and *absorption coefficient* σ_a according to the two possible events of elastic scattering and absorption:

$$\sigma_t(\vec{x}) = \sigma_s(\vec{x}) + \sigma_a(\vec{x}).$$

The probability of elastic scattering given that collision happened is the *albedo* of the material:

$$a = \frac{\sigma_s}{\sigma_t}.$$

The positive term in the right side of equation (1) represents *in-scattering*, i.e. the contribution of photons that come from other directions $\vec{\omega}'$ and get scattered to direction $\vec{\omega}$. The probability that elastic scattering happens in unit distance is σ_s. The probability density of the reflection direction is defined by *phase function* $P(\cos\theta)$ that depends on the cosine of the scattering angle $\cos\theta = \vec{\omega}'\cdot\vec{\omega}$. In order to consider all incident directions, the contributions should be integrated for all directions $\vec{\omega}'$ of the directional sphere Ω.

In *isotropic* (also called diffuse) scattering the reflected radiance is uniform and the phase function is constant. For *anisotropic* scattering, the phase function varies with the scattering angle. The extent of anisotropy is usually expressed by the mean cosine of the scattering angle:

$$g = \int_\Omega (\vec{\omega}'\cdot\vec{\omega})P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega'.$$

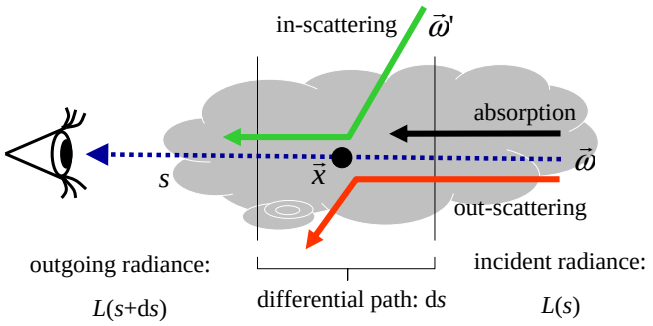


Fig. 1. Change of radiance in participating media.

In *homogeneous media*, volume properties σ_t and σ_s do not depend on position $\vec{x}$. In *inhomogeneous media* these properties depend on the actual position.

The primary source of the illumination may be the surfaces, light sources, or the volume itself. These can be taken into account by either adding a source term to the right side of equation (1) or by enforcing boundary conditions making the radiance of the volume equal to the prescribed radiance of the source.

In case of measured data, material properties are usually stored in a 3D voxel grid, and are assumed to be constant or linear between voxel centers. Let Δ be the distance of the grid points. The total extinction of a voxel can be expressed by the *opacity*:

$$\alpha = 1 - e^{-\sigma_t\Delta} \approx \sigma_t\Delta. \quad (2)$$

Radiance $L(\vec{x},\vec{\omega})$ is often expressed as the sum of two terms, the *direct term* L_d that represents unscattered light, and the *media term* L_m that stands for the light component that scattered at least once:

$$L(\vec{x},\vec{\omega}) = L_d(\vec{x},\vec{\omega}) + L_m(\vec{x},\vec{\omega}).$$

The direct term is reduced by absorption and out-scattering:

$$\frac{\mathrm{d}L_d}{\mathrm{d}s} = -\sigma_t(\vec{x})L_d(\vec{x},\vec{\omega}). \quad (3)$$

The media term is not only reduced by absorption and out-scattering, but also increased by in-scattering:

$$\frac{\mathrm{d}L_m}{\mathrm{d}s} = -\sigma_t(\vec{x})L_m(\vec{x},\vec{\omega}) +$$

$$\sigma_s(\vec{x})\int_\Omega \left(L_d(\vec{x},\vec{\omega}') + L_m(\vec{x},\vec{\omega}')\right)P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega'.$$

Note that this equation can be re-written by considering the reflection of the direct term as a *volumetric source*:

$$\frac{\mathrm{d}L_m}{\mathrm{d}s} = -\sigma_t(\vec{x})L_m(\vec{x},\vec{\omega}) +$$

$$\sigma_s(\vec{x})\int_\Omega L_m(\vec{x},\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega' + \sigma_s(\vec{x})Q(\vec{x},\vec{\omega}), \quad (4)$$

where the source intensity is:

$$Q(\vec{x},\vec{\omega}) = \int_\Omega L_d(\vec{x},\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega'. \quad (5)$$

2.2 γ-photons

The elastic scattering of γ-photons is described by the *Klein-Nishina* formula [2], which expresses the *differential cross section*, i.e. the product of the energy dependent phase function and the scattering coefficient:

$$\sigma_s(\vec{x}, E_0)P(\cos\theta, E_0) = \frac{r_e^2}{2}C(\vec{x})(\epsilon + \epsilon^3 - \epsilon^2 \sin^2\theta),$$

where $\epsilon = E_1/E_0$ expresses the ratio of the scattered photon's energy E_1 and the incident photon's energy E_0, $r_e = 2.82 \cdot 10^{-15}$ [m] is the classical electron radius, and $C(\vec{x})$ is the electron density (number of electrons per unit volume) at point $\vec{x}$.

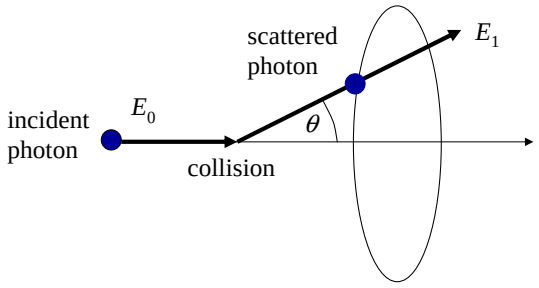


Fig. 2. Compton scattering. The relative energy change E_1/E_0 is determined by the scattering angle θ.

When elastic scattering happens, there is a unique correspondence between the scattered photon's energy and the cosine of the scattering angle, as defined by the *Compton formula*:

$$E_1 = \frac{E_0}{1 + \frac{E_0}{m_e c^2}(1 - \cos\theta)},$$

where $m_e c^2$ is the electron's energy expressed by the product of its rest mass m_e and the square of the speed of light c (Fig. 2). Note that the energy change is relevant when E_0 is non negligible with respect to the energy of the electron. This is not the case for photons of visible wavelengths, when $E_0 \ll m_e c^2$, thus $E_1 \approx E_0$. In this case, the Klein-Nishina phase function becomes similar to the phase function of Rayleigh scattering.

The absorption coefficient can also be expressed as the product of the electron density and a factor that depends on the the photon's energy and the material compounds, and is usually defined by a simple polynomial fitted to measurements.

Due to the coupling of different frequencies, we cannot consider the transport equation on different photon energy levels independently. Instead, a single equation describes the transport on all levels:

$$\left.\frac{\mathrm{d}L(\vec{x} + \vec{\omega}s, \vec{\omega}, E_1)}{\mathrm{d}s}\right|_{s=0} = -\sigma_t(\vec{x}, E_1)L(\vec{x}, \vec{\omega}, E_1) + \int_\Omega L(\vec{x}, \vec{\omega}', E_0)\sigma_s(\vec{x}, E_0)P(\vec{\omega}' \cdot \vec{\omega}, E_0)\mathrm{d}\omega'. \quad (6)$$

In this equation incident photon energy E_0 is not an independent variable since the Compton formula relates it to scattered photon energy E_1 and the cosine of the scattering angle.

Note that equation (1) governing light photons and equation (6) differ only in the frequency on which the incident radiance is taken when the directional integral is evaluated. In case of light transport, the incident radiance of the same frequency should be used. For γ-photons, the frequency or photon energy is the function of the scattered energy and the angle of scattering. This means that most of the methods developed for light photons can also be applied for γ-photons. For example, we can decompose the radiance to direct and media terms, and can also introduce the volumetric source. The equation of the direct term is the same as for light photons since extinction does not change the frequency. Special care should be practiced only when the directional integrals of the media term (equation (4)) and of the volumetric source (equation (5)) are evaluated.

3 The Proposed Method

3.1 Motivation and objectives

In this paper we propose a method that renders multiple scattering effects in large volumes of inhomogeneous media. We examine the particularly important case of point sources. We shall assume that the point light source has *radiant intensity* (i.e. the power per solid angle) Φ_0 and is in the origin of our coordinate system. More complex light sources can be modeled by translation and superposition.

In order to get close to interactive rates, the solution method should be implemented on the GPU. We have taken the iterational approach because of its better convergence rate.

Substituting the finite element approximation, the transport equation and the projection into the finite element basis simplify to a system of linear equations:

$$\mathbf{L} = \mathbf{T} \cdot \mathbf{L} + \mathbf{Q}^e, \quad (7)$$

where vector $\mathbf{L}$ is the radiance of the sample locations and directions, $\mathbf{Q}^e$ is the vector of the finite element representation of the source term and boundary conditions, and $\mathbf{T}$ is the transport matrix.

Iteration obtains the solution as the limiting value of the following iteration sequence:

$$\mathbf{L}_n = \mathbf{T} \cdot \mathbf{L}_{n-1} + \mathbf{Q}^e. \quad (8)$$

To reduce the algorithmic complexity, i.e. to limit the number of interactions considered in a single iteration step, we follow a discrete ordinates like iteration scheme, where the transport matrix elements are obtained on the fly with a very simple approximation. However, the high number of finite elements needed to represent the direction dependent radiance of each voxel still poses performance and storage problems. To get both sufficient

computational power and storage, a GPU cluster is an appropriate choice [22].

The discrete ordinates scheme alone would not be satisfactory because of the following reasons:

- The iteration of local interactions would be slow because it requires many "warming up" steps to distribute the power of sources to far regions.
- Artifacts due to the discretization of the directional domain would show up at regions where the source is highly concentrated, which could be reduced by increasing the number of partitions of the directional sphere, but that would also dramatically increase the storage requirements and decrease the computational speed.
- The scalability of the algorithm would be poor since exchanging data between the computation nodes after each iteration step has significant overhead.

To solve the problems of warming up and discretization artifacts, we *bootstrap* the iteration phase, i.e. we approximate the solution in a cheap way then let the iteration focus on only the "rest" of the problem. Suppose that during bootstrapping we find initial guess $\mathbf{L}_0$ for the finite element representation of the radiance function. The unknown radiance is then expressed as the sum of the initial guess $\mathbf{L}_0$ and some unknown *residual* $\Delta\mathbf{L}$. Substituting this decomposition into the iteration formula of equation (8), we obtain a similar iteration scheme for the residual:

$$\Delta\mathbf{L}_n = \mathbf{T} \cdot \Delta\mathbf{L}_{n-1} + \Delta\mathbf{Q}^e$$

where

$$\Delta\mathbf{Q}^e = \mathbf{T} \cdot \mathbf{L}_0 + \mathbf{Q}^e - \mathbf{L}_0$$

is the source term of the residual. Note that if the initial guess is accurate, than the source of the residual is small. Furthermore, if $\mathbf{L}_0$ extends farther than the direct term determining the volumetric source, then the source of the residual will be more uniform than the source of the original problem. Delivering radiance globally helps the iteration to exchange non-zero radiance even at the beginning of the process. On the other hand, the more uniform source distribution reduces the discretization problem.

To solve the problem of expensive data exchanges in each iteration step, we propose an iteration scheme that exchanges data less frequently. This slows down the convergence of the iteration, so computing nodes should work longer, but reduces the communication load, providing a flexible compromise according to the actual computation and communication speeds.

The outline of the proposed method is shown by Fig. 3. We use a simple and fast technique to initially distribute the light in the medium. The distribution is governed by the diffusion theory, where the single pass approximate solution is made possible by assumptions that the medium is locally homogeneous and spherically symmetric. Note that unlike previous approaches solving the diffusion equation, we do not consider the diffusion approximation as the final solution, but we use it as the initial value of an iteration scheme. Consequently, we can take a different trade off between the accuracy of our diffusion solver and its speed, thus our diffusion solution is more approximate but can be obtained at the same cost as the direct term. Having obtained the initial approximation, the final solution is computed by iteration on a GPU cluster.

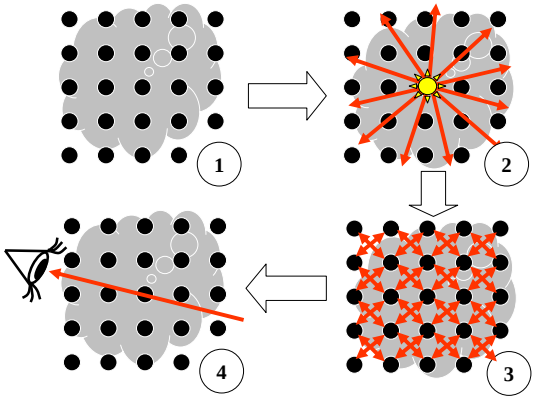


Fig. 3. The outline of the algorithm. 1: The volume is defined by a voxel grid. 2: Single scattering and estimated multiple scattering are distributed from the light source. 3: The final results are obtained by iteration which corrects the errors of the initial estimation. 4: The image is rendered by standard alpha blending.

3.2 Initial approximation

Now we present how an initial approximation can be obtained, which can bootstrap the iteration or a stochastic solution.

Similarly to the diffusion approximation [9] we approximate the direction dependence of the radiance by the first two terms of the spherical harmonic expansion:

$$L(\vec{x},\vec{\omega}) \approx \tilde{L}(\vec{x},\vec{\omega}) = \frac{1}{4\pi}\phi(\vec{x}) + \frac{3}{4\pi}\vec{E}(\vec{x}) \cdot \vec{\omega},$$

where constant term $\phi(\vec{x})$ is called the *fluence* and the coefficient of the linear directional term $\vec{E}(\vec{x})$ is called the *vector irradiance*. By enforcing the equality of the directional integrals of L and $\tilde{L}$, we get the following equation for fluence $\phi(\vec{x})$:

$$\int_\Omega L \mathrm{d}\omega = \int_\Omega \tilde{L} \mathrm{d}\omega \quad \Longrightarrow \quad \phi(\vec{x}) = \int_\Omega L(\vec{x},\vec{\omega})\mathrm{d}\omega.$$

Similarly, requiring the direction weighted averages be similar, we obtain vector irradiance $\vec{E}(\vec{x})$ as:

$$\int_\Omega L\vec{\omega} \mathrm{d}\omega = \int_\Omega \tilde{L}\vec{\omega} \mathrm{d}\omega \quad \Longrightarrow \quad \vec{E}(\vec{x}) = \int_\Omega L(\vec{x},\vec{\omega})\vec{\omega}\mathrm{d}\omega.$$

Substituting this two-term expansion into the radiative transport equation and averaging it for all directions, we obtain the following equations:

$$\vec{\nabla}\phi(\vec{x}) = -3\sigma_t'\vec{E}(\vec{x}), \qquad \vec{\nabla} \cdot \vec{E}(\vec{x}) = -\sigma_a\phi(\vec{x}). \qquad (9)$$

where $\sigma_t' = \sigma_t - \sigma_s g$ is the *reduced extinction coefficient*.

Let us consider just a single beam starting at the origin where the point source is. When a beam is processed, we assume that other beams face the same material characteristics, i.e. we assume that the scene is *spherically symmetric*. Consequently, the solution should also have spherical symmetry. Note that the assumption of spherical symmetry does not mean that only one beam is processed. We take many beams originating from the source, and each of them are traced independently assuming that other rays face the same material properties as the current beam. The advantage of the assumption of spherical symmetry is that we can act as having the complete information of the volume while we march on a single ray. Marching on different rays, different information is used, resulting in a different solution for each ray.

In case of spherical symmetry, the radiance of the inspected beam at point $\vec{x}$ and in direction $\vec{\omega}$ may depend just on distance $r = |\vec{x}|$ from the origin and on the angle between direction $\vec{\omega}$ and the direction of point $\vec{x}$. The fluence depends just on distance r and vector irradiance $\vec{E}(\vec{x})$ has the direction of the given point, that is $\vec{E}(\vec{x}) = E(r)\vec{\omega}_{\vec{x}}$.

Expressing the divergence operator in spherical coordinates, we get:

$$\vec{\nabla} \cdot \vec{E}(\vec{x}) = \vec{\nabla} \cdot (E(r)\vec{\omega}_{\vec{x}}) = \frac{1}{r^2}\frac{\partial(r^2 E(r))}{\partial r}.$$

Thus, the scalar version of equation (9) is:

$$\frac{\mathrm{d}\phi(r)}{\mathrm{d}r} = -3\sigma_t' E(r), \qquad \frac{1}{r^2}\frac{\mathrm{d}(r^2 E(r))}{\mathrm{d}r} = -\sigma_a \phi(r). \quad (10)$$

If we have a point light source, then this equation has a singularity at $r = 0$. The fluence and the vector irradiance are very large close to the source and very small farther away. As we wish to obtain the solution using finite differences, such high magnitude variance makes the solution proposed in [1] numerically unstable in regions where most of the photons arrive after multiple scattering. To solve this problem, we rewrite the equations to use radiant intensity $\psi = r^2 L$ instead of the radiance L. The first two spherical harmonics terms of the radiant intensity are related similarly to the fluence and the vector irradiance:

$$\psi_0 = r^2\phi, \qquad \psi_1 = r^2 E.$$

Substituting these into the differential equation we obtain:

$$\frac{\mathrm{d}\psi_0(r)}{\mathrm{d}r} = \frac{2}{r}\psi_0 - 3\sigma_t'\psi_1(r), \qquad \frac{\mathrm{d}\psi_1(r)}{\mathrm{d}r} = -\sigma_a\psi_0(r). \quad (11)$$

For homogeneous infinite material, the differential equation can be solved analytically:

$$\psi_0^h(r) = Ae^{-\sigma_e r} r,$$

$$\psi_1^h(r) = \frac{2}{3r\sigma_t'}\psi_0^h(r) - \frac{1}{3\sigma_t'}\frac{\mathrm{d}\psi_0^h(r)}{\mathrm{d}r} = \frac{A}{3\sigma_t'}e^{-\sigma_e r}\left(\sigma_e r + 1\right). \quad (12)$$

where $\sigma_e = \sqrt{3\sigma_a\sigma_t'}$ is the *effective transport coefficient*, and A is an arbitrary constant that should be determined from the boundary conditions. According to equation (12) $\psi_0^h(0) = 0$, thus only the second equation is free at the boundary. Requiring that at $r = 0$ the average radiant intensity is equal to that of the source Φ_0, we can calculate the free parameter as:

$$\psi_1^h(0) = \frac{A}{3\sigma_t'} = \Phi_0 \qquad \Longrightarrow \qquad A = 3\sigma_t'\Phi_0.$$

With equation (11) we established two differential equations that describe the power evolving as we move along a ray started at the origin. These equations can be solved by numerical integration while marching on the ray and taking samples from the material properties.

In order to obtain the initial values of the ray marching, we take the solution for homogeneous material in a few voxel neighborhood of the source, initialize the state variables with the homogeneous solution, and iterate them father away. While ray marching makes steps Δ increasing distance r from the source, material properties σ_t and σ_s are fetched at the sample location, and state variables $\psi_0[i]$ and $\psi_1[i]$ are updated according to the numerical quadrature, resulting in the following formula for step i:

$$\begin{aligned} \psi_0[i] &= \psi_0[i-1]\left(1 + \frac{2\Delta}{r[i]}\right) - 3\sigma_t'[i]\psi_1[i-1]\Delta, \\ \psi_1[i] &= \psi_1[i-1] - \sigma_a[i]\psi_0[i-1]\Delta. \end{aligned} \quad (13)$$

Unfortunately, this method cannot be used to estimate the initial radiance approximation in γ-photon transport because it heavily relies on the diffusion equation that describes energy transport on a single wavelength. However, this is not so crucial as for light photons since the energy of γ-photons decreases rather quickly due to the combined effect of absorption and Compton-scattering, which also reduces the energy level of scattered photons. Thus, for γ-photon transport, the direct term is a fairly good approximation far from the source.

3.2.1 Wavefront tracing

During the initial radiance approximation, rays are cast from the origin and we march along these rays to evaluate the optical depth needed by the direct term and to iterate equation (13). On the parallel GPU architecture, we simultaneously process a set of rays, thus the visited points form *wavefronts* (Fig. 3).

To execute wavefront tracing, the volume is resampled to a new grid that is parameterized with spherical coordinates. A voxel of the new grid with (u, v, w) coordinates represents point

$$(x, y, z) = Rw(\cos\delta\sin\beta, \sin\delta\sin\beta, \cos\beta),$$

where $\delta = 2\pi u$, and $\beta = \arccos(1 - 2v)$, and R is the size of the volume. Note that this parametrization provides uniform sampling in the directional domain. A (u, v) pair encodes the ray direction, while w encodes the distance

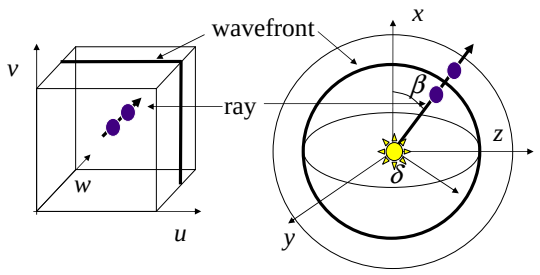


Fig. 4. Wavefront tracing. Marching on all rays starting at the source creates spherical fronts in the original space and planar w-layers in the transformed domain.

from the origin. This texture is processed w-layer by w-layer, i.e. stepping the radius r by Δ simultaneously for all rays.

At the end of wavefront tracing, we return to the original grid and compute the direct term and the approximate radiance of the grid points with tri-linear interpolation.

3.3 Refinement of the initial solution by iteration

As the result of wavefront tracing, we have a good estimate for the direct term:

$$L_d(\vec{x},\vec{\omega}_{\vec{x}}) = \frac{\Phi_0 e^{-\int_0^r \sigma_t(s)\mathrm{d}s}}{r^2},$$

and consequently for the volumetric source:

$$Q(\vec{x},\vec{\omega}) = \int_\Omega L_d(\vec{x},\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega' = L_d(\vec{x},\vec{\omega}_{\vec{x}})P(\vec{\omega}_{\vec{x}}\cdot\vec{\omega}),$$

and a less accurate estimate for the total radiance:

$$L(\vec{x},\vec{\omega}) \approx \frac{\psi_0(r)}{4\pi r^2} + \frac{3\psi_1(r)}{4\pi r^2}(\vec{\omega}_{\vec{x}}\cdot\vec{\omega}).$$

Thus, we can accept direct term L_d, but the media term $L_m = L - L_d$ needs further refinement. We use an iteration scheme to make the media term more accurate, which is based on equation (4), but we consider only the voxel centers to convert the integral equation to a finite system of linear equations. The *incoming medium radiance* arriving at voxel p from direction $\vec{\omega}$ is denoted by $I_m^{(p)}(\vec{\omega})$. Similarly, the *outgoing medium radiance* is denoted by $L_m^{(p)}(\vec{\omega})$. Using these notations, the discretized version of equation (4) at voxel p is:

$$L_m^{(p)}(\vec{\omega}) = (1-\alpha^{(p)})I_m^{(p)}(\vec{\omega}) +$$
$$\alpha^{(p)}a^{(p)}\int_\Omega I_m^{(p)}(\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega' + \alpha^{(p)}a^{(p)}Q^{(p)}(\vec{\omega}) \quad (14)$$

since $\sigma_t\Delta \approx \alpha$ and $\sigma_s\Delta \approx \alpha a$. According to the concept of discrete ordinates, the directional integral is approximated by a finite Riemann-sum requiring the incident radiance just at D sample directions $\vec{\omega}'_1,\ldots,\vec{\omega}'_D$:

$$\int_\Omega I^{(p)}(\vec{\omega}')P(\vec{\omega}'\cdot\vec{\omega})\mathrm{d}\omega' \approx \frac{4\pi}{D}\sum_{d=1}^{D} I^{(p)}(\vec{\omega}'_d)P(\vec{\omega}'_d\cdot\vec{\omega}). \quad (15)$$

In order to speed up the iteration process, here we further simplify the incident radiance estimation, and select the sample directions to be the directions where neighboring voxel centers are visible. This decision makes the incident radiance evaluation trivial, because it will be equal to the outgoing radiance of that voxel whose center is visible in the given direction.

The number of neighbors depends on the structure of the grid. In a conventional *Cartesian Cubic* (CC) grid, a grid point has $D = 6$ neighbors. In a so called *Body Centered Cubic* (BCC) grid [23] a voxel has $D = 8$ neighboring voxels that share a face, which still seems to be too small to approximate a directional integral. Thus, it is better to use a *Face Centered Cubic* (FCC) grid [16], where each voxel has $D = 12$ neighbors (Fig. 5).

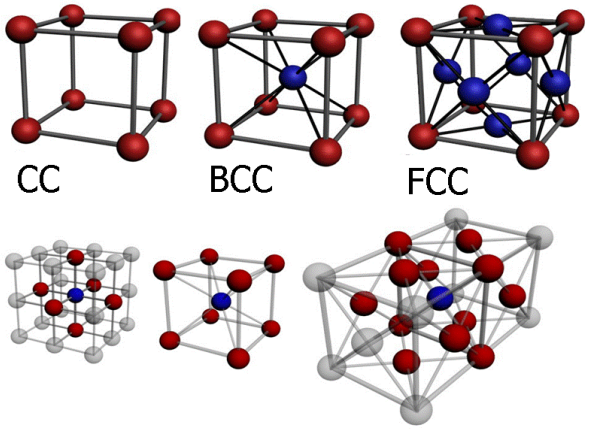


Fig. 5. Structure of the Cartesian, Body Centered, and Face Centered Cubic grids. In a Face Centered Cubic grid, sample points are the voxel corners, voxel centers, and the centers of the voxel faces. Here every grid point has 12 neighbors, all at the same distance.

As the incoming radiance of a voxel equals the outgoing radiance of the neighboring voxel in the given direction, equations (14) and (15) are equivalent to the general linear system of equation (7), where $\mathbf{L}$ is the vector of media radiance values $L_m^{(p)}(\vec{\omega}_d)$ for all voxels $p = 1,\ldots,V$ and discrete directions $d = 1,\ldots,D$, $\mathbf{Q}^e$ is the vector of $\alpha^{(p)}a^{(p)}Q^{(p)}(\vec{\omega}_d)$ products, and matrix $\mathbf{T}$ connects the output radiance of a voxel to the output radiances of its neighboring voxels via their input radiances. Vectors $\mathbf{L}$ and $\mathbf{Q}^e$ have VD elements. Instead of storing sparse matrix $\mathbf{T}$, its elements are computed on the fly by the shader program. The linear system is solved by iteration (equation (8)).

The generalization of the iteration algorithm to γ-photons is straightforward, we just have to store a complete spectrum instead of a radiance value associated with single wavelength. The spectrum is also defined with finite elements, for example, the interesting wavelength range is decomposed to subintervals, and the energy levels associated with the subintervals are

represented by a vector. We describe the spectrum on F pre-defined representative frequencies $[\nu_1, \nu_2, \ldots, \nu_F]$ by a vector $[L_1, L_2, \ldots, L_F]$ where each element represents the intensity of the radiation of photons belonging to an interval around the representative frequency (in the current implementation $F = 4$ and thus the spectrum fits in a single GPU variable).

3.3.1 Iteration on parallel machines

In order to execute iteration on a parallel machine, the radiance vector $\mathbf{L}_n$ of step n is broken to parts and each computing node is responsible for the update of its own part. However, the new value of a part also depends on other parts, which would necessitate state exchanges between the nodes in every iteration step. This would quickly make the communication the bottleneck of the parallel computation.

For example, if there are two compute nodes, the radiance vector and the iteration scheme are decomposed as:

$$\mathbf{L}_n = \begin{bmatrix} \mathbf{L}_n^1 \\ \mathbf{L}_n^2 \end{bmatrix} = \begin{bmatrix} \mathbf{T}^{11} & \mathbf{T}^{12} \\ \mathbf{T}^{21} & \mathbf{T}^{22} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{L}_{n-1}^1 \\ \mathbf{L}_{n-1}^2 \end{bmatrix} + \begin{bmatrix} \mathbf{Q}^{e1} \\ \mathbf{Q}^{e2} \end{bmatrix}.$$

Node 1 would multiply with minor matrices $\mathbf{T}^{11}$ and $\mathbf{T}^{12}$, while node 2 with $\mathbf{T}^{21}$ and $\mathbf{T}^{22}$. After these multiplications, estimate $\mathbf{L}_n^1$ should be moved to node 2 from node 1, and similarly, $\mathbf{L}_n^2$ should be moved to node 1 from node 2. The communication load can be reduced by recognizing that only those elements should be exchanged where the column of minor matrices $\mathbf{T}^{12}$ and $\mathbf{T}^{21}$ are not zero. For example, defining the neighborhood of a voxel by an FCC grid, only a single layer of the 3D texture may directly affect a part of the volume, so only boundary layers should be exchanged. However, even these data transfers get unacceptably slow in comparison to the computational performance of the GPUs.

This problem can be solved by not exchanging the current state in every iteration cycle. Suppose, for example, that we exchange data just in every second iteration cycle. When the data is exchanged before executing the matrix-vector multiplication, the iteration looks like the original formula:

$$\mathbf{L}_n = \mathbf{T} \cdot \mathbf{L}_{n-1} + \mathbf{Q}^e.$$

However, when the data is not exchanged, a part of the transfer matrix is multiplied by the radiance estimate of the older iteration. Let us decompose transport matrix $\mathbf{T}$ to $\mathbf{T}_{\text{own}}$ that has the same matrix elements as $\mathbf{T}$ where the own part is multiplied and to $\mathbf{T}_{\text{other}}$ that stores the matrix elements used by the other node:

$$\mathbf{T}_{\text{own}} = \begin{bmatrix} \mathbf{T}^{11} & 0 \\ 0 & \mathbf{T}^{22} \end{bmatrix}, \qquad \mathbf{T}_{\text{other}} = \begin{bmatrix} 0 & \mathbf{T}^{12} \\ \mathbf{T}^{21} & 0 \end{bmatrix},$$

With this notation, the cycle without previous data exchange is:

$$\mathbf{L}_n = \mathbf{T}_{\text{own}} \cdot \mathbf{L}_{n-1} + \mathbf{T}_{\text{other}} \cdot \mathbf{L}_{n-2} + \mathbf{Q}^e.$$

Putting the two equations together, the execution of an iteration step without state exchanges and then an iteration step with state exchanges would result in:

$$\mathbf{L}_n = \mathbf{T}^2 \cdot \mathbf{L}_{n-2} + \mathbf{T} \cdot \mathbf{Q}^e + \mathbf{Q}^e + \mathbf{T} \cdot \mathbf{T}_{\text{other}} \cdot (\mathbf{L}_{n-3} - \mathbf{L}_{n-2}).$$

Note that if this scheme is convergent, then $\mathbf{L}_n$, $\mathbf{L}_{n-2}$, and $\mathbf{L}_{n-3}$ should converge to the same vector $\mathbf{L}$, thus the limiting value satisfies the following equation:

$$\mathbf{L} = \mathbf{T}^2 \cdot \mathbf{L} + \mathbf{T} \cdot \mathbf{Q}^e + \mathbf{Q}^e.$$

This equation is equivalent to the original equation, which can be proven if the right side's $\mathbf{L}$ is substituted by the complete right side:

$$\mathbf{L} = \mathbf{T} \cdot \mathbf{L} + \mathbf{Q}^e = \mathbf{T} \cdot (\mathbf{T} \cdot \mathbf{L} + \mathbf{Q}^e) + \mathbf{Q}^e.$$

The price of not exchanging the data in every iteration step is the additional error term

$$\mathbf{T} \cdot \mathbf{T}_{\text{other}} \cdot (\mathbf{L}_{n-3} - \mathbf{L}_{n-2}).$$

This error term converges to zero, but slows down the iteration process especially when the iteration is far from the converged state.

Using the same argument, we can prove a similar statement for more than two nodes and for cases when the data is exchanged just in every third, fourth, etc. cycles. The number of iterations done by the nodes between data exchanges should be specified by finding an optimal compromise, which depends on the relative computation and communications speeds.

4 Parallel implementation

The system has been implemented as a parallel application running on a 4 node HP Scalable Visualization Array (SVA), where each node is equipped with an NVIDIA GeForce 8800 GTX GPU, programmed under CUDA. The nodes are interconnected by Infiniband.

The tasks are distributed by subdividing the volume along one axis and each node is responsible for both the radiative transport simulation and the rendering of its associated subvolume. The images rendered by the nodes are composited by the ParaComp library [24].

Wavefront tracing computing the direct term and the initial radiance approximation is fast and needs just a small fraction of the total computation time. Thus, its parallelization is not necessary if the volume fits into a single GPU memory or downsampling the volume for the direct term computation is allowed by the required accuracy. In this case, each node starts wavefront tracing from the source, computes the direct term and the radiance approximation independently of other nodes, and terminates rays when they leave the block associated with this node. Due to the ray termination, even this redundant computation can benefit from the addition of more nodes.

For larger volumes exceeding the memory capacity of a single node, wavefront tracing should also be implemented as a parallel application where communication

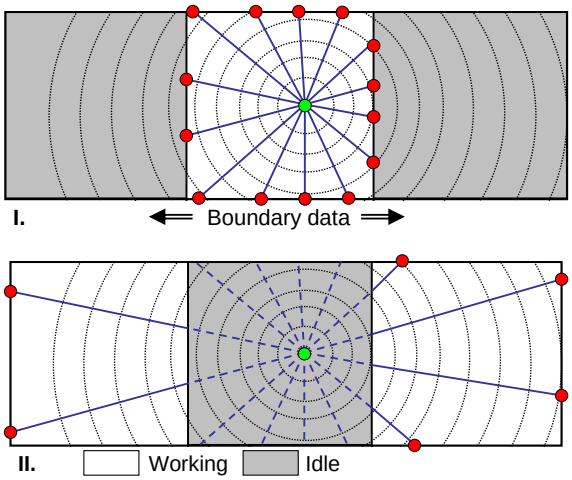


Fig. 6. Parallel implementation of wavefront tracing for three nodes, where the middle node contains the source. Note that in each of the two phases, only a subset of nodes is working.

is also needed between the nodes. As the radiance approximation values depend on previous ray-marching steps, a node can start the wavefront tracing process only if the rays entering its subvolume are known. Fig. 6 illustrates the concept of our implementation. Ray marching begins in the node which holds the source, and proceeds normally until each ray gets terminated. Then the exit values of the rays are stored in a boundary layer, which is a 2D array with the same parametrization as the wavefront texture. This array is sent to the neighboring nodes, which can continue the ray marching in their own subvolume. Note that data parallelization of wavefront tracing allows volumes larger than the capacity of a single node, but it may not keep all nodes always busy.

During the iterational refinement, separate kernels are executed on the GPU for each computational step. The radiance distribution for one wavelength in the FCC grid is represented with 12 floating point arrays — one for each discrete direction in the grid. The FCC sites can be mapped into a standard 3D array by using proper indexing, where each value means the outgoing radiance from a given grid site in one direction. The volumetric source values remain constant during the iteration, so we store them in separate 3D textures. The iteration kernel updates the state of the grid by reading the emissions and the incoming radiances from the neighboring grid sites. The output of an iteration step is the input of the following one, so we copy the results back to the input textures after each kernel execution. In order to improve performance, we introduced a sensitivity constant which is a lower bound to the sum of the incoming radiances for each point. We evaluate the iteration formula only where the radiance value is greater than this constant. This method is efficient if there are larger parts of the volume without significant irradiance.

In addition to executing the iteration in the individual subvolumes, we need to implement the radiance transport between the neighboring volume parts. The simulation areas overlap so that the radiance values at the boundary layer can be seamlessly passed from one subvolume to the other. MPI communication between the nodes is used to exchange the solutions at the boundary layers. It is important to notice that each node needs to pass only 4 arrays to its appropriate neighbor as the FCC grid has 4 outgoing and 4 incoming directions for each axis-aligned boundaries.

Having obtained the view independent solution of the transfer equation, the partial images of the blocks are rendered in parallel with alpha blending, and also composited in a distributed way taking advantage of the parallel pipeline compositing scheme of the ParaComp library.

5 Results

In this section we present our experiments to validate the method and to measure its performance. In order to validate the method, we first put a cube filled with homogeneous isotropic media ($\sigma_t = 5, a = 0.8$) into the empty space ($\sigma_t = 0$), and calculated the initial estimation with the previous [1] and the new methods, the iteration solution, and a Monte Carlo solution for comparison. The cube has edge size 2 and is discretized by a 128^3 resolution voxel grid. The Monte Carlo algorithm is based on photon mapping [15]. The radiance values computed along a line are shown by Fig. 7. Note that the new initial estimation is more stable than the previous one (i.e. it is closer to the reference solution), and also that the error of the initial estimation is quite well compensated by the iteration process.

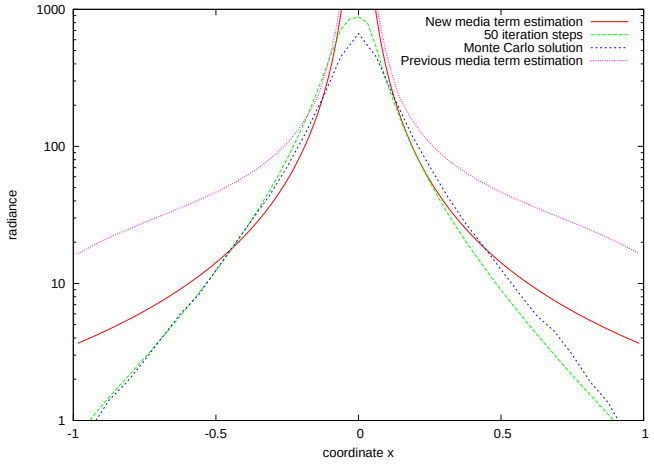


Fig. 7. The radiance functions along line $x \in (-1, 1), y = 0, z = 0$ obtained with the iteration solution, the previous and new initial estimations, and with Monte Carlo photon mapping.

The evolution of the iteration can also be followed and compared to the Monte Carlo reference in Fig. 9, Fig. 10,

and Fig. 11 for the different resolution models of the *Head*, *Beetle*, and the *Visible Human* datasets, respectively. The radiance is color coded to emphasize the differences and is superimposed on the image of the density field. The resolution of the screen is 600×600.

The density field of the Head dataset has 128^3 voxels. The FCC grid has $128 \times 128 \times 64$ resolution. First, we have examined the effect of the initial radiance approximation. Fig. 8 shows the relative L_1 error curves of the iteration obtained when the radiance is initialized by the direct term only and when the media term approximation is also used. Note that the application of the media term approximation halved the number of iteration steps required to obtain a given precision. When we initialize the iteration with the direct term, we need about 100 iteration steps to eliminate any further visual change in the image (the error goes below 2%). However, when the radiance is initialized with the approximated total radiance, we obtain the same result executing only 60 iterations.

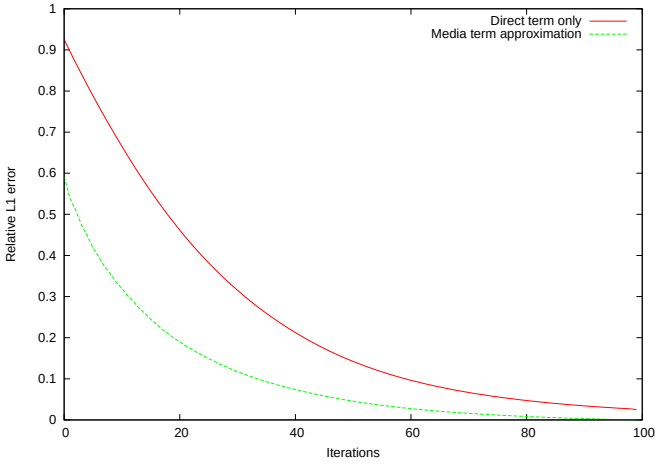


Fig. 8. Relative L_1 error curves of the iteration when the radiance is initialized to the single-scattering term and when the radiance is initialized to the media term approximation in the Head dataset. We used the converged result as a reference. Note that with the initial media term approximation, roughly only 50% of the iteration steps are needed.

No	Initial	Iter	Com (1 : 5)	Visual	Total (1 : 5)
2	27 ms	19 ms	23 : 10 ms	35 ms	2.6 : 1.8 s
3	19 ms	12 ms	25 : 10 ms	30 ms	2.2 : 1.4 s
4	17 ms	8 ms	29 : 11 ms	25 ms	2.1 : 1.1 s

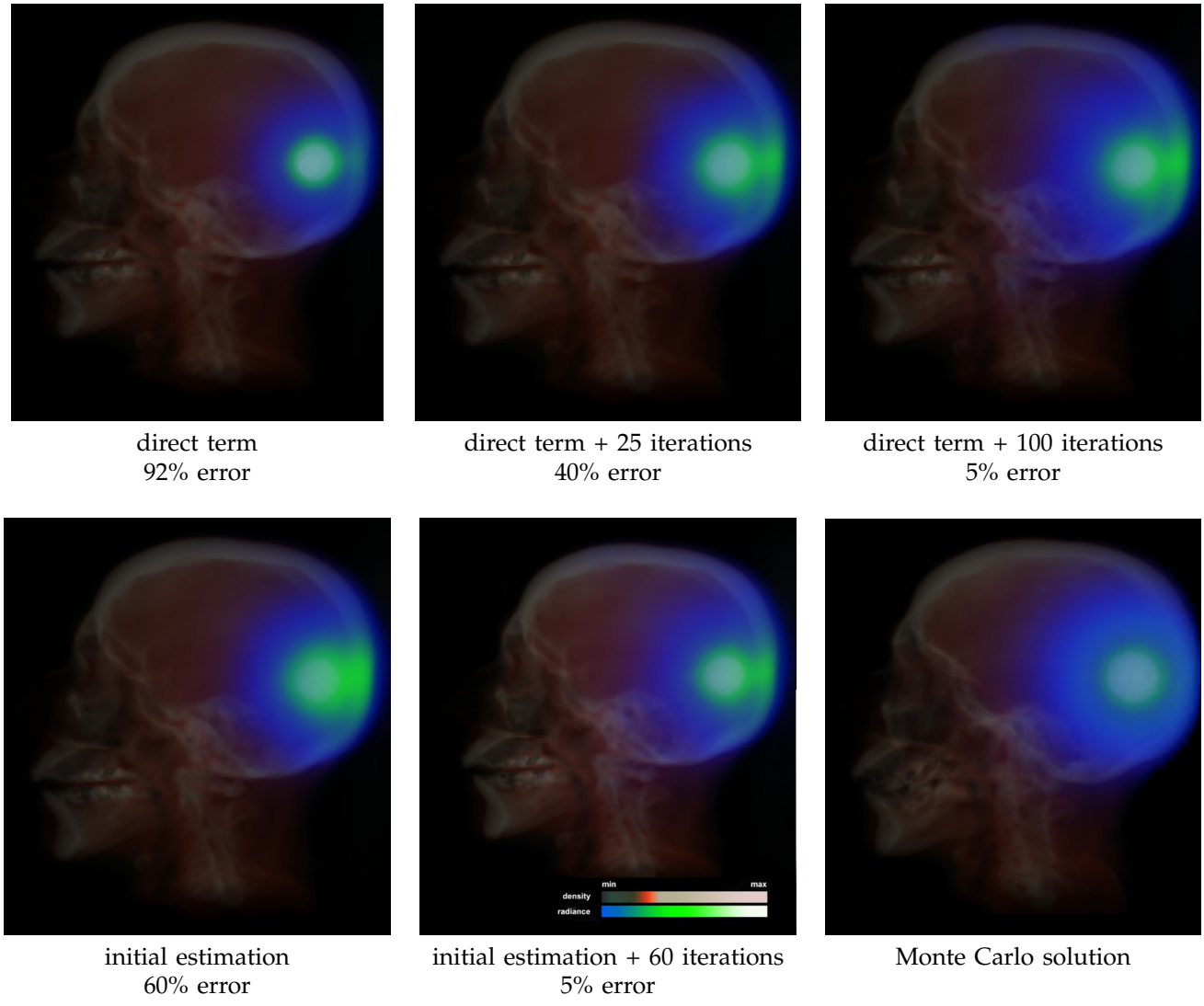
TABLE 2
Performance figures measured for the Head dataset with respect to the number of nodes ("No").

We tested the scalability of our parallel implementation for the Head and Beetle datasets for 2, 3, and 4 nodes (column "No" in Tables 2 and 3). The volume is decomposed into 4 blocks along one of the coordinate axes, and the transfer of each block is computed on a separate node equipped with its own GPU.

Table 2 summarizes the time data measured for the Head dataset (Fig. 9). "Initial" time is needed by the initial radiance distribution. As this model is relatively small, the initial radiance distribution is executed redundantly on all nodes without any communication. Due to early ray termination that stops rays at block boundaries, we can still observe a speed up with the introduction of additional nodes. "Iter" is the computation time of a single iteration cycle on the FCC grid. "Com" is the average time required by the exchanges of the boundary layers and texture ping-pong after the iteration cycle. This time is measured separately for the case when the boundary layers are exchanged in each iteration step and when they are exchanged after every fifth iteration step, and the two time values are separated by a ":" symbol. "Visual" is needed by the final ray casting and compositing the partial images. "Total" includes the total simulation/rendering times needed to reduce the error below 2% when boundary conditions are exchanged in each iteration step (60 iterations) and when boundary conditions are exchanged after every fifth step (63 iterations). As the initial radiance distribution and visualization are executed only once, but iteration and communication times are multiplied by the number of iterations in the total simulation/rendering time, data in the "Iter" and "Com" columns are primarily responsible for the overall performance.

We can observe that the speed of the execution of an iteration step ("Iter") grows super-linearly with additional nodes. The explanation is that additional nodes not only increase the computational power, but by reducing the data size of a single node, they also improve cache utilization. However, despite the reduced iteration time, the total simulation/rendering time improves with additional nodes just moderately when boundary layers are exchanged in each iteration step because of the communication bottleneck ("Com" becomes larger than "Iter" and does not decrease with the introduction of new nodes). This bottleneck can be eliminated by exchanging the boundary conditions less frequently, which slightly reduces the speed of convergence, so we trade communication overhead for GPU computation power. We observed that the error caused by exchanging the boundary conditions just after every fifth iteration cycle can be compensated by about 5% more cycles, which is a good tradeoff (we executed 63 iteration cycles instead of 60). Note that when we exchanged the boundary conditions just after every fifth iteration cycle, the total simulation/rendering speed increased to 160 % when the number of nodes has been increased from two to four.

Performance data obtained with the Beetle dataset (Fig. 10) are shown by Table 3 for different FCC grid resolutions ("Resolution"). The density field is defined by $416 \times 416 \times 247$ voxels. Note that the highest resolution FCC grid can be processed only with four nodes. In this



direct term
92% error

direct term + 25 iterations
40% error

direct term + 100 iterations
5% error

initial estimation
60% error

initial estimation + 60 iterations
5% error

Monte Carlo solution

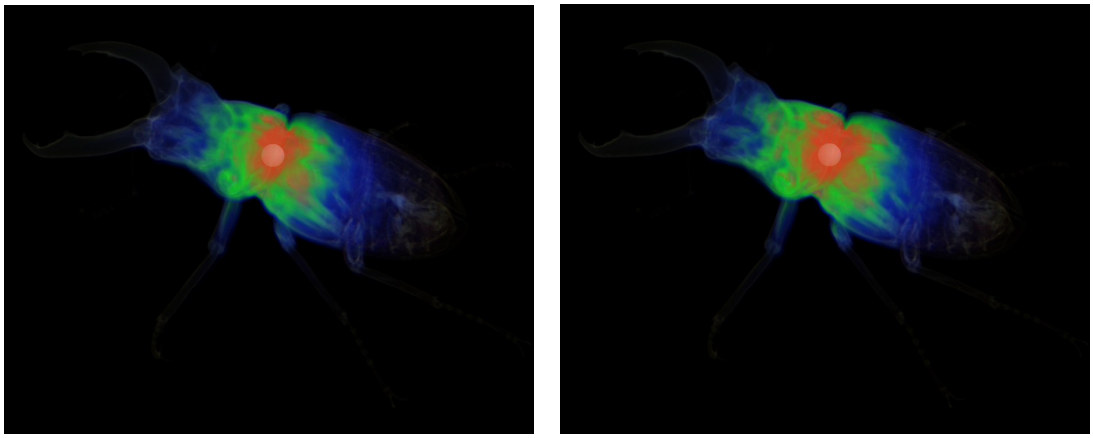
Fig. 9. Evolution of the iteration in the Head dataset when the radiance is initialized to the direct term and to the estimated media term, respectively. We also included the relative L_1 error values obtained with respect to the Monte Carlo reference. Note that in this example the media term estimation is equivalent to the direct term estimation followed by 18 iteration steps.

Resolution	No	Initial	Iter	Com (1 : 5)
$256 \times 256 \times 76$	2	120 ms	63 ms	57 : 44 ms
	3	81 ms	38 ms	60 : 32 ms
	4	61 ms	30 ms	60 : 24 ms
$336 \times 336 \times 100$	3	181 ms	100 ms	117 : 70 ms
	4	130 ms	66 ms	114 : 55 ms
$384 \times 384 \times 112$	4	194 ms	91 ms	127 : 76 ms

TABLE 3
Performance figures measured for the Beetle dataset with respect to resolution of the FCC grid ("Resolution") and the number of nodes ("No").

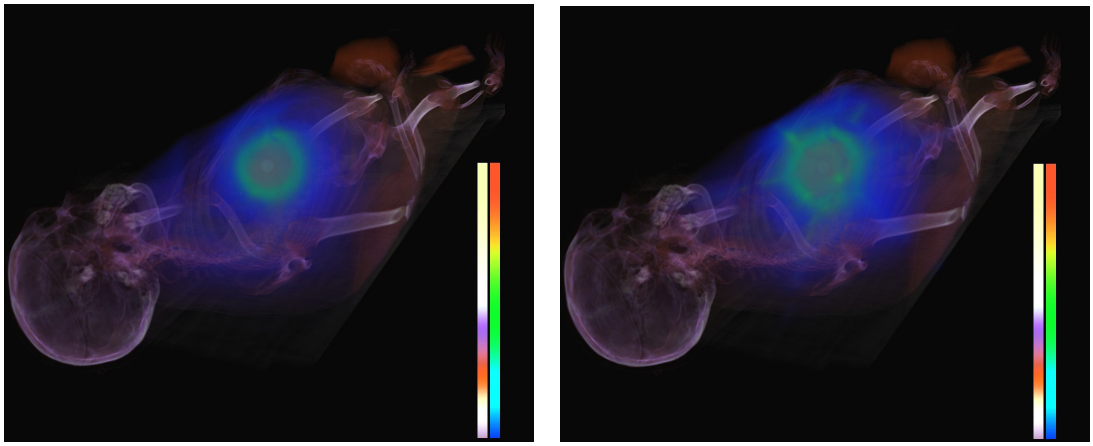
case, the initial radiance distribution is also a parallel application that exchanges results. The visualization step needed less than 50 ms for all cases. When the boundary is exchanged just after every fifth cycle, the total computation times for the smallest resolution — including the initial radiance distribution, 60 iteration steps that guarantee convergence, and the visualization — are 6, 4, and 3.5 seconds on 2, 3, and 4 nodes, respectively. The largest resolution model can be rendered in 10 seconds on 4 nodes.

Fig. 11 demonstrates that the method can be scaled to as large volumes as the Visible Human. Using 4 nodes, the initial estimation required 177 ms, a single iteration without communication needed 136 ms, and with communication 244 ms. With respect to the head model, here we reduced the probability of absorption to increase the distance that can be reached by the radiation and to allow the examination of the problems of directional discretization. The figure compares the iteration results



initial estimation (56% error) initial estimation + 50 iterations (6% error)

Fig. 10. Radiance transport in the Beetle dataset. The density field is defined by $416 \times 416 \times 247$ voxels, and the resolution of the FCC grid is $384 \times 384 \times 112$. In this example, the media term approximation could also save about 30 iteration steps.



initial estimation + 30 iterations (5% error) direct term + 60 iterations (13% error)

Fig. 11. Radiance transport in the Visible Human dataset. The resolution of the density field is $512 \times 512 \times 1877$ and the resolution of the FCC grid is $200 \times 200 \times 368$. This model can only be simulated on four nodes because of the memory requirements. Note that if we start the iteration with the proposed initial radiance approximation (left), the spurious rays, i.e. the ray effect, due to the directional discretization get reduced with respect to the iteration starting with the direct term only (right). The left and right bars depict the transfer functions of the extinction coefficient and the radiance, respectively.

obtained when we start with the proposed initial approximation and when we start with the direct term only. As the initial approximation smoothes the volumetric source term even at far regions, the new method will be less sensitive to the directional discretization, thus the ray effect artifacts that are clearly visible in the right image can be significantly reduced.

The accuracy of γ-photon transport is also examined using the cube phantom filled with homogeneous media ($C(\vec{x}) = 10/r_e^2, a = 0.8$). We assume that γ-photons are produced by positron-electron annihilation events as in PET, thus the initial photon energy is 511 keV.

As γ-photon scattering is not isotropic, we depict the directional average of the radiance on three frequency ranges in Fig. 12. The frequency ranges are [112–212] keV, [212–312] keV, and [312–512] keV. During iteration, the photons are assigned to these frequency ranges in every iteration step. On the other hand, the Monte Carlo reference solution calculates the photon energy levels accurately during photon tracing, and projects the final result into these ranges just in the final visualization step. Note that the γ-photon simulation is less accurate than the simulation of light photons. The reasons are the additional finite element representation of the fre-

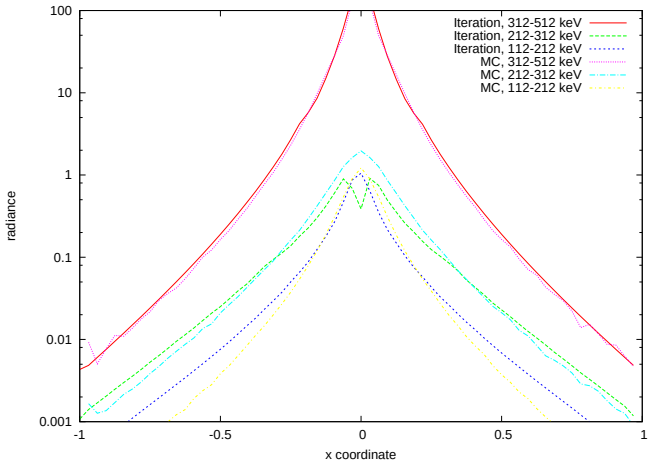


Fig. 12. The average intensity of the γ-radiation along line $x \in (-1, 1), y = 0, z = 0$ on three photon energy levels, obtained with the iteration solution and with Monte Carlo photon mapping.

quency range that introduces some error in each iteration step, and the specular-shaped lobe of the Klein-Nishina phase function that poses problems to the FCC grid's directional discretization. We can also observe an energy drop for the second frequency range close to the source, which is due to the fact that at such high initial photon energy level, after a scattering event the new photon energy may be in between the 33% (back-scattering) and the 100% (forward-scattering) of the original energy. The 12 discrete directions and the 4 discrete energy levels cannot accurately handle such high dynamic range. Thus, if more accurate results are needed, the method is recommended for lower photon energy levels, used, for example, in SPECT (140 keV) and X-ray or gamma-ray brachytherapy (20-40 keV or 380 keV).

Fig. 13 shows the results of γ-photon transport in the head dataset where the initial photon energy is 140 keV, i.e. we assume a typical SPECT source. The examined photon energy range is decomposed to four intervals, [60–80] keV, [80–100] keV, [100–120] keV, and [120–140] keV, thus the spectrum is stored in a single float4 variable. The electron density is scaled to make the bone density equal to $10/r_e^2$. In this simulation, only the direct term is approximated before the iteration is started.

Concerning the performance of γ-photon transport, we observed that the initial distribution, the iteration, and the visualization are just slightly slower than the light photon simulation. However, the communication time increased significantly since light photon simulation exchanges float textures, but γ-photon simulation should work with float4 textures representing the whole spectrum. The calculation of Fig. 13 with 20 iteration steps required 2 seconds on four nodes.

6 Conclusions

This paper proposed an effective method to solve the radiative transport equation in inhomogeneous participating media on a cluster of GPUs, allowing interactive source placement since the solution is obtained in a few seconds. The final results are provided by an iterational solver based on an FCC grid. The iterative algorithm has been significantly improved by finding a good initial guess for the radiance and modifying the parallel implementation to reduce the frequency of data exchanges. Without these, the discretization artifacts would be unacceptable, and the very high performance of GPUs would make the communication become the bottleneck. This concept of iterating more on the nodes without exchanges gives us a versatile tool to address the scalability issue. We have tested the approach on a cluster of GPUs, but it is equally applicable to multiple GPU cards inserted in the same desktop since they also share the problem of the communication bottleneck.

Acknowledgement

This work has been supported by the National Office for Research and Technology, Hewlett-Packard, OTKA K-719922 (Hungary), and by the Teratomo project.

References

[1] L. Szirmay-Kalos, G. Liktor, T. Umenhoffer, B. Tóth, S. Kumar, and G. Lupton, "Parallel solution to the radiative transport," in *Eurographics Symposium on Parallel Graphics and Visualization*, Comba, Debattista, and Weiskopf, Eds., 2009, pp. 95–102.

[2] C. N. Yang, "The Klein-Nishina formula & quantum electrodynamics," *Lect. Notes Phys.*, vol. 746, pp. 393–397, 2008.

[3] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion, "A survey on participating media rendering techniques," *The Visual Computer*, vol. 21, no. 5, pp. 303–328, 2005.

[4] J. F. Blinn, "Light reflection functions for simulation of clouds and dusty surfaces," in *SIGGRAPH '82 Proceedings*, 1982, pp. 21–29.

[5] B. Sun, R. Ramamoorthi, S. G. Narasimhan, and S. K. Nayar, "A practical analytic single scattering model for real time rendering," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1040–1049, 2005.

[6] V. Pegoraro and S. G. Parker, "An Analytical Solution to Single Scattering in Homogeneous Participating Media," *Computer Graphics Forum (Proceedings of the 30th Eurographics Conference)*, vol. 28, no. 2, pp. 329–335, 2009.

[7] M. Harris and A. Lastra, "Real-time cloud rendering," *Computer Graphics Forum*, vol. 20, no. 3, pp. 76–84, 2001.

[8] J. Kniss, S. Premoze, C. Hansen, and D. Ebert, "Interactive translucent volume rendering and procedural modeling," in *VIS '02: Proceedings of the conference on Visualization '02*, 2002, pp. 109–116.

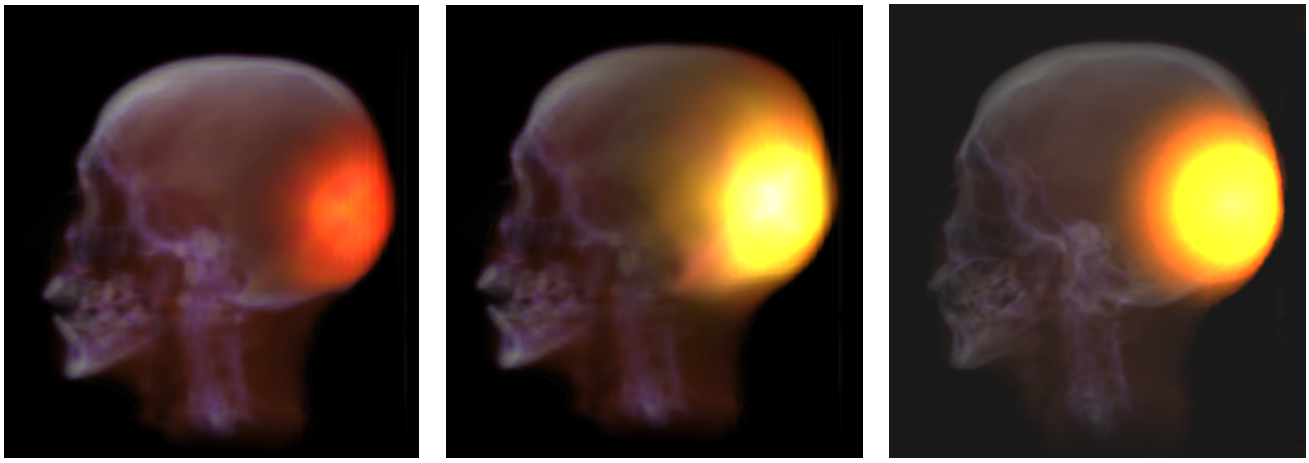
[9] J. Stam, "Multiple scattering as a diffusion process," in *Eurographics Rendering Workshop*, 1995, pp. 41–50.

[10] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *SIGGRAPH '01 Proceedings*, 2001, pp. 511–518.

[11] X. Tong, J. Wang, S. Lin, B. Guo, and H.-Y. Shum, "Modeling and rendering of quasi-homogeneous materials," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 1054–1061.

[12] R. Geist, K. Rasche, J. Westall, and R. J. Schalkoff, "Lattice-boltzmann lighting," in *Rendering Techniques*, 2004, pp. 355–362.

[13] T. Haber, T. Mertens, P. Bekaert, and F. Van Reeth, "A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects," in *GI '05: Proceedings of Graphics Interface*, 2005, pp. 79–86.



direct term | direct term + 20 iterations | MC reference

Fig. 13. Gamma photon transport in the head dataset with color coded energy levels ([120–140] keV: red, [100–120] keV : green, [80–100] keV : blue). Note that around the source the radiation is strong at all frequencies (the color is white), but only higher energy photons survive Compton scattering farther, which makes the color yellow and finally red.

[14] J. Wang, S. Zhao, X. Tong, S. Lin, Z. Lin, Y. Dong, B. Guo, and H.-Y. Shum, "Modeling and rendering of heterogeneous translucent materials using the diffusion equation," *ACM Trans. Graph.*, vol. 27, no. 1, pp. 1–18, 2008.

[15] H. W. Jensen and P. H. Christensen, "Efficient simulation of light transport in scenes with participating media using photon maps," *SIGGRAPH '98 Proceedings*, pp. 311–320, 1998.

[16] F. Qiu, F. Xu, Z. Fan, and N. Neophytos, "Lattice-based volumetric global illumination," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1576–1583, 2007, fellow-Arie Kaufman and Senior Member-Klaus Mueller.

[17] K. Zhou, Z. Ren, S. Lin, H. Bao, B. Guo, and H.-Y. Shum, "Real-time smoke rendering using compensated ray marching," *ACM Trans. Graph.*, vol. 27, no. 3, p. 36, 2008.

[18] L. Szirmay-Kalos, M. Sbert, and T. Umenhoffer, "Real-time multiple scattering in participating media with illumination networks," in *Eurographics Symposium on Rendering*, 2005, pp. 277–282.

[19] J. Kajiya and B. V. Herzen, "Ray tracing volume densities," in *Computer Graphics (SIGGRAPH '84 Proceedings)*, 1984, pp. 165–174.

[20] H. E. Rushmeier and K. E. Torrance, "The zonal method for calculating light intensities in the presence of a participating medium," in *SIGGRAPH '87 Proceedings*, 1987, pp. 293–302.

[21] R. Fattal, "Participating media illumination using light propagation maps," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1–11, 2009.

[22] M. Strengert, M. Magallon, D. Weiskopf, S. Guthe, and T. Ertl, "Hierarchical visualization and compression of large volume datasets using gpu clusters," in *Proceedings of EUROGRAPHICS Symposium on Parallel Graphics and Visualization*, 2004, pp. 41–48.

[23] B. Csébfalvi, "Prefiltered gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid," in *VIS '05: Visualization, 2005*. IEEE Computer Society, 2005, pp. 311–318.

[24] Paracomp, "Hp scalable visualization array version 2.1," HP, Tech. Rep., 2007, http://docs.hp.com/en/A-SVAPC-2C/A-SVAPC-2C.pdf.

László Szirmay-Kalos is the head of Department of Control Engineering and Information Technology at the Budapest University of Technology and Economics. He received Ph.D. in 1992 and full professorship in 2001 in computer graphics. His research area is Monte-Carlo global illumination algorithms and their GPU implementation. He has more than two hundred publications in this field. He is the fellow of Eurographics.

Gábor Liktor was graduated from the Budapest University of Technology and Economics. His diploma work was a collaboration with Spinor Gmbh and dealt with computer games. He currently works on GPGPU algorithms and special effects in games, and has started his Ph.D. studies at the University of Stuttgart recently.

Tamás Umenhoffer is an assistant professor at the Budapest University of Technology and Economics. His research topic is the computation of global illumination effects and realistic lighting in participation media, and their application in real-time systems and games.

Balázs Tóth is an assistant professor at the Budapest University of Technology and Economics. He is involved in distributed GPGPU projects and deferred shading rendering, and is responsible for the CUDA education of the faculty.

Shree Kumar works for Hewlett-Packard. He is the main developer of the ParaComp compositing library. His interests include the creation of large GPU clusters and real-time parallel graphics applications.

Glenn Lupton is a senior member of the technical staff in the High-Performance Computing Division at Hewlett-Packard, where he was the leader of visualization, compositing, and GPGPU developments. He has been the invited speaker of the EG Symposium of Parallel Graphics and Visualization, and conferences on supercomputing.