Real-Time Multiple Scattering using Light Propagation Volumes

Markus Billeter Erik Sintorn Ulf Assarsson
Chalmers University of Technology*

(a) *Rendering with our new method*

(b) *Single Scattered Volumetric Shadows*

(c) *Light Propagation Volumes*

Figure 1: *Real-time rendering produced by our new method (a) in comparison to single-scattered volumetric shadows (b) and light propagation volumes (c). Our method captures indirect illumination by light scattered in a participating medium and captures higher-order scattering effects in the participating medium. The view in (a) is rendered at* $\sim$ 30 *FPS at a resolution of* 1280×720 *by our implementation on an NVIDIA GTX480 GPU using OpenGL and CUDA.*

Abstract

This paper introduces a new GPU-based, real-time method for rendering volumetric lighting effects produced by scattering in a participating medium. The method includes support for indirect illumination by scattered light, high-quality single-scattered volumetric shadows, and approximate multiple scattered volumetric lighting effects in isotropic and homogeneous media. The method builds upon an improved propagation scheme for light propagation volumes. This scheme models scattering according to the radiative light transfer equation during propagation. The initial state of the light propagation volumes is based on single-scattered light identified with shadow maps; this allows generation of a high quality initial distribution of radiance. After propagation, the resulting distribution is used as a source of diffuse light during rendering and is also ray marched for volumetric effects from multiple scattering. Volumetric shadows from single-scattered light are rendered separately. We compare the new method to single-scattered volumetric shadows produced by contemporary techniques, plain light propagation volumes (which this new method extends), and a simple composition thereof.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: real time, scattering, light propagation volumes

1 Introduction

Scattering of light in a participating medium produces volumetric lighting effects that add important visual cues to renderings, improve depth perception and produce more realistic images. Recent real-time methods, such as those presented in Section 2, have focused on providing volumetric shadows considering single-scattering only.

While volumetric shadows belong to the more visually impressive effects produced by scattering, real-time methods generally fail to include several other important effects, such as indirect surface illumination by the scattered light and higher-order scattering effects in the participating medium.

For instance, outside of the shafts of light produced by the volumetric shadows algorithms, one has to rely on other methods (such as the ad-hoc distance fog) to emulate the presence of a participating medium. Figure 1b demonstrates this: outside of the shafts of light the medium is completely invisible. Our proposed method, shown in Figure 1a, handles this case. Note that no explicit ambient light has been introduced. Surfaces not affected by direct light are il-

*e-mail: {billeter,d00sint,uffe}@chalmers.se

luminated indirectly by scattered light (and in some cases by light reflected from nearby objects).

Sun et al. [2005] treated indirect surface illumination from single-scattered light; however, their semi-analytic solution does not take blocking geometry into account. Techniques like photon mapping can simulate indirect illumination by scattered light (with blocking geometry taken into account), but are often unsuitable for real-time rendering of complex dynamic scenes, especially if higher-order scattering effects are considered.

The method presented in this paper augments contemporary real-time algorithms with support for indirect illumination by scattered light from both single and multiple scattering with high-quality single-scattered and approximate multiple-scattered volumetric shadows. Higher-order scattering effects are approximated using a modified variant of the light propagation volume (LPV) algorithm [Kaplanyan and Dachsbacher 2010]. Since LPVs perform badly for high-frequency effects, such as the sharp volumetric shadows from single-scattering, these are rendered separately.

The current implementation assumes that the participating medium is isotropic and homogeneous, although this limitation could be relaxed in the future. As no data is kept between frames, our method supports fully dynamic scenes.

2 Related Work

Scattering in participating media is treated by a vast number of papers. This section is limited to recent contributions and papers that present concepts important to the work presented in this paper. Cerezo et al. [2005] present an introduction to and summary of works older than 2005.

Scattering Model and Light Transport. In the model described by Cerezo et al., light is attenuated by out-scattering and absorption in accordance with Beer's law:

$$I(x) = I_0\, e^{-\sigma_t\, x}.$$

The extinction coefficient, σ_t, is related to the optical density of a homogeneous participating medium, and x is the distance at which the intensity $I(x)$ is computed, given an initial intensity I_0. Beer's law is derived from the differential equation $dI(x) = -\sigma_t\, I(x)\, dx$, i.e. at each point in space a fraction of the intensity $I(x)$ is absorbed or scattered away [Chandrasekhar 1960].

From the model above, the air-light integral [Nishita et al. 1987] can be derived for a point light source. The air-light integral describes the amount of light that is scattered once on its path from a light source to the observer. Since solving the air-light integral efficiently is central to many single-scattering methods, many different methods have been developed, such as the semi-analytic solution with a lookup texture by Sun et al. [2005] or the closed form solutions by Pegoraro et al. [2009; 2010; 2011].

The differential form of Beer's Law is also an essential part of the transport equations describing radiative light transfer:

$$\omega \cdot \nabla L\left(\mathbf{x}, \omega\right) = -\sigma_t L\left(\mathbf{x}, \omega\right) + \sigma_s \int p\left(\omega, \omega'\right)\; L\left(\mathbf{x}, \omega'\right)\; d\omega', \tag{1}$$

where ω represents a direction; $\mathbf{x}$ a position in space; $p\left(\omega, \omega'\right)$ is the phase function; and $\sigma_t = \sigma_a + \sigma_s$ is the extinction coefficient, computed from the absorption and scattering coefficients [Arvo 1993; Geist et al. 2004]. Radiance, described by $L\left(\mathbf{x}, \omega\right)$, replaces the one-dimensional intensity $I(x)$ in Beer's Law. In addition to the extinction modeled by Beer's Law, in-scattering is introduced in the transport equation.

Volumetric Shadows. Many papers treat rendering volumetric shadows using the air-light model. A popular approach is to approximatively solve the air-light integral using various ray marching methods, such as iteration using fragment shaders [Dobashi et al. 2002; Imagire et al. 2007]. Optimizations include sharing data between pixels [Toth and Umenhoffer 2009; Engelhardt and Dachsbacher 2010]. Recent work by Baran et al. [2010] accelerates the ray marching by constructing an acceleration structure from a shadow map. Chen et al. [2011] extend this work by adding support for textured light sources.

Wyman and Ramsey [2008] use shadow volumes to limit the range where ray marching is required. The ray marching is eliminated by Billeter et al. [2010] by using fast solutions to the airlight integral to integrate between planes in the shadow volume. Wyman [2011] instead voxelizes the space into an epipolar-space grid with binary visibility for fast lookups.

Most real-time methods handle isotropic and homogeneous media only. One exception is Zhou et al. [2007], who approximate heterogeneous media, such as smoke clouds, with Gaussian base functions. The approach is later extended to include multiple scattering [Zhou et al. 2008].

Our method renders the high-frequency volumetric shadows dominated by single-scattering separately. Treating single-scattering separately from higher order terms has previously successfully been used, e.g. in the context of subsurface light transport [Jensen et al. 2001]. Although any method for rendering single-scattered volumetric shadows could be used, our implementation uses the method by Billeter et al. [2010]. This method relies on plain shadow maps only, which are reused in other stages of the algorithm.

Light Propagation Volumes. Kaplanyan and Dachsbacher [2010] present light propagation volumes (LPVs) as a method to capture indirect illumination between surfaces. The method operates on a lattice, where each cell c stores a radiance $L_c\left(\omega\right)$. This radiance is propagated to neighboring sites using the discrete ordinate method [Chandrasekhar 1960], where propagation occurs into a fixed number of directions. Spherical Harmonics (SHs) efficiently represent and store the radiance at each lattice site [Evans 1998].

As our new method relies on and extends the LPV algorithm, we summarize it here. Radiance from a cell c to its neighbor c'_j, which lies in direction j from c, is transferred by projecting the cell's radiance L_c on a transfer function Γ_j. This extracts the radiance $\Delta L_{c'_j}$ that is propagated in the direction from cell c to its neighbor c'_j:

$$\Delta L_{c'_j} = \langle L_c | \Gamma_j \rangle\; \Gamma_j.$$

Evaluation of the dot product, denoted $\langle \cdot | \cdot \rangle$, becomes very efficient when using spherical harmonics coefficients to represent radiances and transfer functions.

An initial distribution of radiance is generated from reflective shadow maps (RSMs) [Dachsbacher and Stamminger 2005]. Each texel is injected into the LPV lattice to give a small contribution of radiance in the direction of the surface normal at that texel. Additionally, Kaplanyan and Dachsbacher [2010] implement fuzzy blocking by injecting an approximate representation of scene geometry reconstructed from shadow maps into a blocker lattice. The blocker lattice contains functions $\mathcal{B}_{cc'_j}\left(\omega\right)$, also represented using

(a) *Surface illumination only* **(b)** *With ray marching* **(c)** *Our method*

Figure 2: *Light propagation volumes (LPVs) used for surface illumination only (a). Ray marching the LPV gives an in-air glow close to the illuminated surface (b), but it is unclear why and where this glow originates. Adding single-scattering components gives the viewer a clear hint that the surface is illuminated from a light source residing inside the object at the center of the screen (c). Note: all pictures use the modified propagation algorithm, i.e., scattering is taken into account during propagation and ray-marching.*

SH coefficients, which describe the approximate amount of occlusion between two cells. The complete propagation scheme can be summarized as

$$\begin{aligned} b_{cc'_j} &= \left\langle \mathcal{B}_{cc'_j} \middle| \Gamma_j \right\rangle \\ \Delta L_{c'_j} &= (1 - b_{cc'}) \ \langle L_c | \Gamma_j \rangle \ \Gamma_j, \end{aligned} \tag{2}$$

where $b_{cc'_j}$ expresses the amount of blocking between c and c'_j.

We modify this scheme to include scattering during propagation in Section 3. Our modified method provides an approximate solution to the radiative light transfer equation (Equation (1)), and resembles the method presented by Geist et al. [2004], who rely on a similar setup but store separate coefficients for each propagation direction.

Each iteration of the propagation is also accumulated into a separate lattice, which, after a sufficient number of propagation iterations, forms the LPV that is used to illuminate the scene.

Kaplanyan and Dachsbacher [2010] introduce limited scattering effects by ray marching the LPV. The ray marching captures light reflected by surfaces in mid-air. However, as light sources produce no radiance in the LPV, it is unclear where the light originates – it looks as if the surface itself may be emitting light. Figure 2 demonstrates this.

3 Propagation Method

In this section, we extend the propagation method in light propagation volumes to account for scattering. As we use dedicated methods to render single-scattered volumetric shadows, we must ensure that our modifications allow isolation of single-scattering from higher-order scattering effects.

Radiance transported from one cell to its neighbors is affected by two phenomena [Chandrasekhar 1960; Arvo 1993]:

1. Extinction, consisting of out-scattering and absorption, reduces the radiance arriving at the neighbor.
2. In-scattering increases the radiance.

Extinction between cells c and c'_j is controlled by the factor $\lambda = d\,\sigma_t$, calculated from the extinction factor σ_t and the average distance d between two neighboring cells. The transferred radiance is simply reduced by a factor λ. In an isotropic medium, in-scattering adds radiance proportional to the total out-scattered radiance from

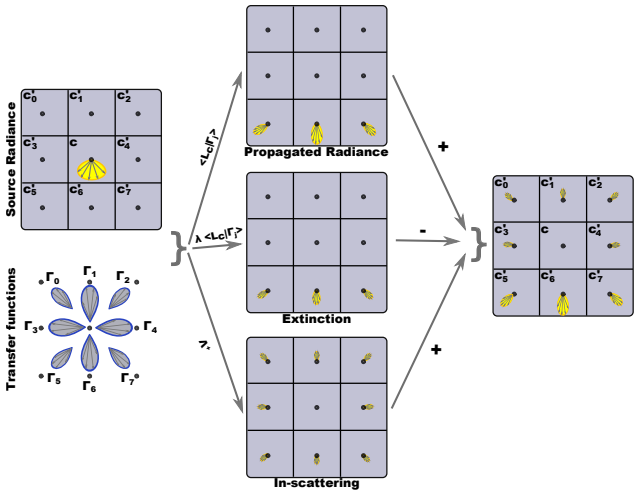


Figure 3: *Illustration of propagation scheme described by Equation (3). Initially, only cell c contains radiance (top left). Using the transfer functions Γ_j (bottom left), radiance is then propagated from c to its neighbors c'_j. The propagation consists of three components: transferred radiance (top middle), extinction (center) and in-scattering (bottom middle). Summing these contributions yield the radiance distribution (right side). This radiance distribution is scaled using the blocking factors $b_{cc'_j}$ (not shown in figure). Without scattering, the cells c'_0 through c'_4 would never receive any radiance.*

the current cell. The in-scattered radiance $\Lambda^+_{cc'_j}$ can be calculated as follows:

$$\begin{aligned} \gamma_j &= \frac{\langle \Gamma_j | \mathbf{1} \rangle}{\sum_l \langle \Gamma_l | \mathbf{1} \rangle} \\ \Lambda^+_{cc'_j} &= \gamma_j \ \lambda_s \ \langle L_c | \mathbf{1} \rangle . \end{aligned}$$

For cell c, the total energy is given by $\langle L_c | \mathbf{1} \rangle \equiv \int_\Omega L_c \, d\omega$, which, multiplied with $\lambda_s = d\,\sigma_s$, measures the total out-scattered energy from the cell ($\mathbf{1}$ denotes the function $f(\omega) = 1$). A fraction, γ_j, of the total out-scattered energy is added to the radiance transferred to c'_j; this fraction is equal to the ratio of radiance captured by the transfer function between cells c and c'_j.

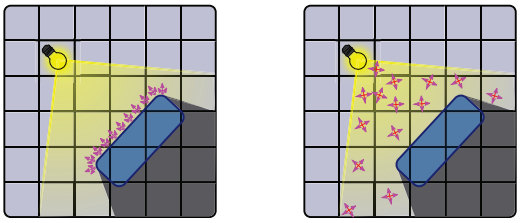
Combining all these factors yields the following:

$$\Delta L_{c'_j} = \left(1 - b_{cc'_j}\right) \left((1 - \lambda) \ \langle L_c | \Gamma_j \rangle + \Lambda^+_{cc'_j} \right) \Gamma_j . \tag{3}$$

The blockers $b_{cc'_j}$ are computed as described in Equation (2). Figure 3 illustrates the propagation scheme described by Equation (3);

Although already heavily discretized, the propagation scheme presented in Equation (3) still resembles the transfer equation presented in Equation (1). The expression $\langle L_c | \Gamma_j \rangle$ extracts the radiance for a subset of ω that coincide with direction j. Thus, the expressions $\lambda \ \langle L_c | \Gamma_j \rangle$ and $\sigma_t \, L(\mathbf{x}, \omega)$ both describe extinction. Similarly, $\Lambda^+_{cc'_j} = \gamma_j \lambda_s \int_\Omega L_c \, d\omega$ and $\sigma_s \int p(\omega, \omega') \ L(\mathbf{x}, \omega') \ d\omega'$ both describe in-scattering, although at this point we have chosen to limit ourselves to an isotropic medium, and therefore $p(\omega, \omega')$ is equal to a constant.

The transfer functions Γ_j must be chosen carefully (see Section 4), as to ensure that propagation conserves energy (in the absence of blockers and with $\sigma_a = 0$).



(a) *Reflected light* **(b)** *Single scattered light*

Figure 4: *Initialization of the radiance distribution. Light reflected by illuminated surfaces is injected as directed radiance, with the direction determined from the surface normal (a). Kaplanyan and Dachsbacher [2010] describe this process in detail. Additionally, single-scattered light is injected (b). Single scattered light is found in directly illuminated regions of space, which are identified using shadow maps. Radiance constructed from single-scattered light is omnidirectional.*

3.1 Initial Radiance Distribution

In addition to the modified propagation scheme, we need to create an appropriate initial radiance distribution with which we seed the LPV. A simple approach would be to inject light sources and propagate light from these, but this approach has several drawbacks. First, we need to potentially perform a large number of propagation steps for light to reach all parts of the scene. Second, with low resolution lattices, the errors become large.

We instead inject reflected light from reflective shadow maps (RSMs), as described by Kaplanyan and Dachsbacher [2010] (Figure 4a). In addition to this, radiance produced by single-scattering is also injected into the LPV (Figure 4b) using information from a standard shadow map. Since we do not wish to include single-scattered light in the LPV, the injected radiance from single-scattering should only be considered during the first propagation step. When rendering the RSMs, radiance stored in the RSM is attenuated according to Beer's Law in order to account for the participating medium.

4 Implementation

In the previous section, we presented the modified propagation scheme, intentionally omitting implementation-specific details (as far as possible). This section documents the implementation details and choices made in our proof-of-concept implementation. We use OpenGL-shaders for most operations, with the exception of the propagation, which is implemented using CUDA.

Our choices mostly follow those presented by Kaplanyan and Dachsbacher [2010]. For instance, we represent and store functions of the type $f(\omega)$ using spherical harmonics coefficients for the first two bands (for a total of four coefficients). Each cell in the LPV lattice stores three such functions, one for each RGB-color channel.

In our implementation, we consider the nearest 26 neighbors in a cubic lattice during propagation. The 26 transfer functions Γ_j and the related coefficients γ_j are pre-computed offline, as they only depend on the lattice configuration. The transfer functions are represented using spherical harmonics coefficients. As the functions overlap slightly, i.e., $\langle \Gamma_j | \Gamma_i \rangle \neq 0$, when approximated using only two SH bands, we rescale the functions to ensure that the total energy is conserved during propagation. In tests with no blockers or absorption, less than 1% energy is lost in each iteration.

We note that other choices are possible, and that the scheme presented in Section 3 is still valid for many such choices. Unless otherwise noted, the lattices have a resolution of 32^3. We do not store any data between frames (with the exception of the transfer functions Γ_i and the related coefficients γ_i which only depend on the chosen lattice configuration) — each frame, we generate a new initial radiance distribution, render new blockers, and perform the propagation.

4.1 Injection of the Initial Radiance Distribution

To generate the initial radiance distribution, we inject both single-scattered radiance computed from a plain shadow map and reflected radiance from computed from the reflective shadow map. When injecting single-scattered radiance, we take advantage of the fact that we limit ourselves to isotropic media. In this case, the function $f_i(\omega)$ describing the radiance of the sample to be injected is constant on the sphere, and therefore only the first SH coefficient is non-zero. We use this to efficiently accumulate and temporarily store single-scattered radiance in a single *RGB(A)* floating point texture.

Single-scattering only occurs in the sub-space directly illuminated by the light source. We find this space from a standard shadow map. In our implementation, we generate up to eight samples along each ray connecting the light source to a shadow map texel. The samples are randomly distributed along the ray. Samples from the reflective shadow map are injected at the appropriate places in the lattice. The radiance is computed by constructing a smooth SH representation from the normal, multiplied by the color of the RSM texel. This step requires storage of the full $4 \times 3 = 12$ coefficients stored in three *RGBA* floating point textures.

4.2 Propagation

Each frame, prior to propagation, we pre-calculate the 26 blocking factors $b_{cc'_j}$ for each cell in the lattice. This reduces the number of computations during propagation, at the cost of memory usage. Transfer functions Γ_j and constants such as γ_j are placed in CUDA constant memory.

Each propagation step consists of one kernel launch, where each cell is assigned to a thread. The thread visits each of the 26 neighbors in turn and accumulates its new $L_c^{i+1} = \sum_j \Delta L_{c_j}^i$, according to Equation (3).

Scalar products $\langle a|b \rangle$ in Equation (3) are evaluated as a standard 4-vector dot product of the spherical harmonics coefficients. Similarly, integrals of the form $\langle a|\mathbf{1} \rangle$ can be efficiently evaluated using a single multiplication of the first SH coefficient with $2\sqrt{\pi}$.

We have to calculate a separate L_c^{i+1} for each color channel. Each color channel uses three buffers: two hold temporary states between propagation steps, and the third accumulates radiance from all propagation steps. After propagation, the third buffer holds the final radiance distribution used during rendering.

4.3 Rendering

We render geometry using a standard lighting model, where the final radiance distribution from the propagation contributes additional diffuse light. Each fragment determines in which cell of the LPV it is located and samples the radiance in that cell using a surface normal.

Ray marching captures the illumination of the participating medium in a separate post-processing stage. The current implementation simply casts a ray for each pixel originating at the eye position and terminating at the depth indicated by the depth buffer (or when leaving the LPV). The ray samples the LPV at a pre-determined density, and attenuates the sampled radiance based on the distance from the eye according to Beer's law. After ray-marching, we render single-scattered light additively to the resulting image.

5 Results and Evaluation

We tested our implementation on an NVIDIA GTX 480 GPU, rendering at a resolution of 1280×720. Unless otherwise noted, the Crytek Sponza scene is used for measurements. Like Kaplanyan and Dachsbacher [2010], we use reflective shadow maps of resolution 256^2 and eight propagation iterations for a 32^3 LPV lattice. The depth buffer component of the reflective shadow map is reused for injection of single-scattered radiance and as input to the method by Billeter et al [2010] that is used to render high-frequency single-scattered light. We render six additional shadow maps, from which the fuzzy blockers are computed.

Propagation performance is largely independent of the scene and view, whereas injection performance is fairly dependent on the view. Table 1 presents performance for both injection and propagation. For comparison, we have also included timings for a 64^3 grid. Propagation performance is, as expected, heavily affected by grid resolution. On the other hand, injection performance is relatively constant with respect to this. Injection performs badly when many samples are injected into the same cells, with a worst case of around 7.5ms (this occurs when the light is in a small enclosed volume). Injecting blockers takes the largest slice of time here; a better, scene-dependent, choice of (fewer) blocker-shadow maps could reduce this significantly.

Ray marching is currently the most time consuming step, taking approximately 14 ms at full resolution. The ray marching primarily tries to capture low frequency effects. It is therefore possible to ray march at a lower resolution and up-sample the results without major loss of visual quality. At half resolution, the cost for ray marching is reduced to around 4 ms.

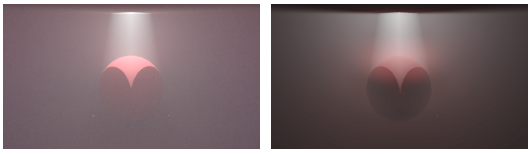
Forward rendering of the Sponza scene with additional diffuse light from the LPV adds around 1.7 ms compared to using only the standard lighting model in our implementation ($\sim$ 3 ms in total). We render single-scattered light volumes in less than 2 ms.

Figure 5 displays a rendering generated with our method and compares it to results provided by plain LPVs, plain single-scattered volumetric shadows, and a trivial combination thereof. Our proof-of-concept implementation renders the view shown in Figure 5a at 32 FPS using full resolution ray marching and at 47 FPS using half resolution ray marching. The same view, using a 64^3 grid, renders at 18 FPS and 25 FPS, respectively.

Figures 7 and 8 display comparisons to ground truth images, rendered offline using LuxRender [LuxRender 2011]. While differences are quite visible, many of the important scattering phenomena are present in the renderings from our method. LuxRender uses a non-trivial tone-mapping operation to create the final image.

Table 1: *Injection and propagation performance for 32^3 and 64^3 grids. Injection consists of three distinct steps: radiance injection using reflective shadow maps, blocker injection, and finally injection of single-scattered radiance. For propagation, we measure four distinct numbers: initialization where input is aggregated from textures into CUDA buffers; preparation of the $b_{cc'_j}$ blocker factors; the actual eight iterations of propagation; and, finally, transfer of the results from CUDA buffers to textures. For each measurement we list the average time and the min/max times observed for different view and light configurations. Included in the total propagation time, but not specifically listed below, is a constant overhead from cudaGraphicsMapResources (~ 0.5 ms for both 32^3 and 64^3).*

	Grid = 32^3	*Grid* = 64^3
Total - Injection	5.9 ms min: 4.5, max: 7.6	5.2 ms min: 3.7, max: 7.1
reflected	0.87 *ms* *min:* 0.59, *max:* 1.1	0.84 *ms* *min:* 0.63, *max:* 1.0
blockers	4.5 *ms* *min:* 3.4, *max:* 6.5	3.9 *ms* *min:* 2.6, *max:* 5.2
single-scattered	0.47 *ms* *min:* 0.39, *max:* 0.55	0.58 *ms* *min* = 0.40, *max:* 0.81
Total - Propagation	2.5 ms min: 2.4, max: 2.6	17.0 ms min: 16.8, max: 17.6
aggregate input	0.02 *ms* *min:* 0.02, *max:* 0.02	0.16 *ms* *min:* 0.15, *max:* 0.16
prepare blockers	0.40 *ms* *min:* 0.40, *max:* 0.40	2.8 *ms* *min:* 2.7, *max:* 2.8
propagate $8\times$	1.49 *ms* *min:* 1.49, *max:* 1.50	12.3 *ms* *min:* 12.2, *max:* 12.5
copy results	0.18 *ms* *min:* 0.18, *max:* 0.19	1.24 *ms* *min:* 1.36, *max:* 1.21



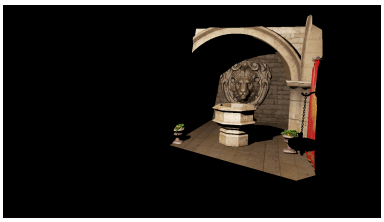
(a) *Offline Reference Image* **(b)** *Our method*

Figure 7: *Comparison to offline rendered reference image. The important scattering phenomena are present in our rendering. Most obviously missing is some illumination at the ceiling, possibly due to overblocking from the fuzzy blockers. Rendering of the reference image took more than 10 hours on a quad core Intel Core2 Q9300.*

(a) *Offline Reference Image* **(b)** *Our method*

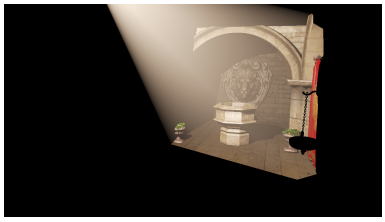
Figure 8: *Comparison of our method to an offline rendering of the test scene. We have translated materials from the original scene to the more advanced material model used by LuxRender. This has introduced some disparities, such has missing bump-maps and vegetation. More indirect illumination is visible in the reference image. We believe this to be connected to how the point source and its immediate surroundings are handled.*

(a) *Our Method*



(b) *Plain Shadow Map*

(c) *Plain LPVs*

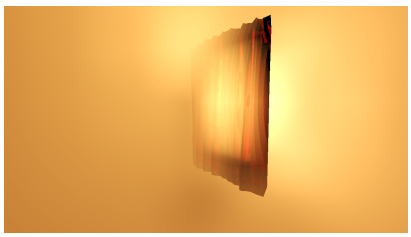


(d) *Single Scattering Only*

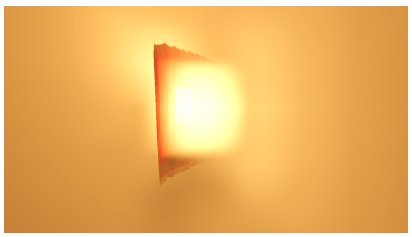
(e) *LPV with Ray Marching*

(f) *Trivial combination of (d) and (e)*

Figure 5: *Comparisons between various techniques and combinations thereof. A reference image is shown in (b), which is rendered using shadow mapping and standard lighting only. Adding single-scattered volumetric shadows (d) enables the viewer to locate the light source, but no additional surfaces in the scene are illuminated. Using plain light propagation volumes (c) adds some indirect lighting around the already lit space in the scene - however, again most of the scene remains hidden. Ray marching plain LPVs yields bloom-like highlights close to illuminated surfaces (e). A trivial combination of LPVs with ray marching and single-scattering (f) gives some additional visual cues about the scene, but again no new surfaces become visible. However, injecting single-scattered light into the LPV adds illumination to previously hidden surfaces and makes the participating medium fully visible (a). Image (a) is rendered by our implementation at approximately 33 FPS using full resolution ray marching. Note: brightness of the images has been increased for printing purposes.*



(a) *Light Bleeding through thin objects*



(b) *Extreme Initial Radiance Distribution*

(c) *Single Propagation Iteration*

Figure 6: *Demonstration of several problematic cases. Light bleeding is shown in (a), where a strong light source is directed at the far side of a thin object. The near side, visible by the camera, is affected by light bleeding through the object. When the camera is placed on the other side of the same object (b), the problem with extreme initial distributions of radiance becomes visible. In (c), a single propagation iteration is taken.*

In contrast, our current implementation simply renders the various components (single-scattered light, multiple scattered light and geometry) in separate steps, and performs a trivial additive composition.

5.1 Limitations

There are of course several limitations to this method. The coarse voxelization of the scene presents several problems, such as light bleeding and "blockiness". The former is mainly visible in scenes with many thin separating features (e.g., thin walls) that cut through voxels. Figure 6a demonstrates this effect. Light bleeding is mentioned by Kaplanyan and Dachsbacher [2010], who suggest that their cascaded LPV approach reduces bleeding to some degree. We have not implemented the cascaded method, but would expect similar improvements.

Blockiness is caused by large gradients in the LPV lattice. The main causes of large gradients are extreme initial radiance distributions and insufficient propagation. Extreme initial distributions are somewhat tricky to deal with, as they break the assumption that the LPVs are used to simulate low-frequency effects. Figure 6b demonstrates the problem by aiming a strong light source at a small surface area (the surface area is completely contained within a voxel).

Increasing the number of propagation iterations can also reduce blockiness at the cost of performance, although large fuzzy blocking may interfere with the smoothing from extra propagation. In Figure 6c we perform only one propagation iteration to illustrate the problem. With the suggested eight propagation iterations we do not observe this problem in the absence of large fuzzy blocking.

Besides the coarse spatial approximation, we also roughly approximate spherical functions by using only the first two spherical harmonics bands. Adding more bands allows more accurate function representation, at the cost of memory consumption and bandwidth (three bands require 9 coefficients, compared to the four used currently). Also, in our limited tests with three bands, ringing/aliasing artifacts in the approximated functions became more of a problem when compared to using just two bands.

We currently only treat homogeneous and isotropic participating media. Although we believe that the propagation scheme can be improved to include support for heterogeneous and anisotropic media, it is unclear how well the low resolution voxelization used in this paper would work with this. Also, real-time single-scattering methods that support heterogeneous and anisotropic media would be required to render the high-frequency single-scattered volumetric shadows.

6 Conclusion and Future Work

We presented a new method for rendering scattering effects in a participating medium. This includes an improved propagation scheme for light propagation volumes; this new scheme supports a participating medium. Additionally, we have suggested basing the initial radiance distribution in the LPV on single-scattered light, which increases quality and reduces the required number of propagation steps. Single-scattered light is treated separately during rendering, enabling independent rendering of high-frequency volumetric shadows with specialized methods. We compare our method to plain single-scattered volumetric shadows, the light propagation volumes presented by Kaplanyan and Dachsbacher [2010], and a trivial combination of these techniques.

In our renderings, no explicit ambient light is required. Rather, light from light sources is scattered by the medium (and performs a single bounce against directly illuminated geometry), which then illuminates surfaces indirectly. Similarly, our method naturally includes fog without resorting to ad-hoc methods.

Since the lattice is relatively coarse, light leaks despite fuzzy blockers, as described by Kaplanyan and Dachsbacher [2010]. A better method for modelling blocking could improve correctness and quality significantly. For instance, by adaptively increasing grid resolution, blocking and propagation quality could be improved. Additionally, one could investigate different packing structures where cells have fewer and/or more equidistant neighbors, which would improve performance and quality, respectively.

Acknowledgements

We would like to thank Sally McKee for her help with the language in this paper and the reviewers for their helpful suggestions.

References

ARVO, J. 1993. Transfer equations in global illumination. In *Global Illumination, SIGGRAPH '93 Course Notes*.

BARAN, I., CHEN, J., RAGAN-KELLEY, J., DURAND, F., AND LEHTINEN, J. 2010. A hierarchical volumetric shadow algorithm for single scattering. In *ACM SIGGRAPH Asia 2010 papers*, 178:1–178:10.

BILLETER, M., SINTORN, E., AND ASSARSSON, U. 2010. Real time volumetric shadows using polygonal light volumes. In *Proceedings of the Conference on High Performance Graphics*, Eurographics Association, HPG '10, 39–45.

CEREZO, E., PEREZ-CAZORLA, F., PUEYO, X., SERON, F., AND SILLION, F. 2005. A survey on participating media rendering techniques. *the Visual Computer 21*, 5, 303–328.

CHANDRASEKHAR, S. 1960. *Radiative transfer*. Dover books on physics and chemistry. Dover Publications.

CHEN, J., BARAN, I., DURAND, F., AND JAROSZ, W. 2011. Real-time volumetric shadows using 1d min-max mipmaps. In *Symposium on Interactive 3D Graphics and Games*, ACM, I3D '11, 39–46.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM, I3D '05, 203–231.

DOBASHI, Y., YAMAMOTO, T., AND NISHITA, T. 2002. Interactive rendering of atmospheric scattering effects using graphics hardware. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, 99–107.

ENGELHARDT, T., AND DACHSBACHER, C. 2010. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM, 119–125.

EVANS, K. F. 1998. The Spherical Harmonics Discrete Ordinate Method for Three-Dimensional Atmospheric Radiative Transfer. *Journal of Atmospheric Sciences 55* (Feb.), 429–446.

GEIST, R., RASCHE, K., WESTALL, J., AND SCHALKOFF, R. J. 2004. Lattice-boltzmann lighting. In *Proceedings of the 15th Eurographics Workshop on Rendering Techniques, Norköping, Sweden, June 21-23, 2004*, Eurographics Association, 355–362.

IMAGIRE, T., JOHAN, H., TAMURA, N., AND NISHITA, T. 2007. Anti-aliased and real-time rendering of scenes with light scattering effects. *Vis. Comput. 23*, 9, 935–944.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '01, 511–518.

KAPLANYAN, A., AND DACHSBACHER, C. 2010. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, 99–107.

LUXRENDER, 2011. Gpl physically based renderer. `http://www.luxrender.net/en_GB/index`.

NISHITA, T., MIYAWAKI, Y., AND NAKAMAE, E. 1987. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '87, 303–310.

PEGORARO, V., AND PARKER, S. G. 2009. An Analytical Solution to Single Scattering in Homogeneous Participating Media. *Computer Graphics Forum 28*, 2 (april), 329–335.

PEGORARO, V., SCHOTT, M., AND PARKER, S. G. 2010. A closed-form solution to single scattering for general phase functions and light distributions. *Computer Graphics Forum (Proceedings of the 21st Eurographics Symposium on Rendering) 29*, 4, 1365–1374.

PEGORARO, V., SCHOTT, M., AND SLUSALLEK, P. 2011. A mathematical framework for efficient closed-form single scattering. In *Proceedings of Graphics Interface 2011*, Canadian Human-Computer Communications Society, GI '11, 151–158.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S. G., AND NAYAR, S. K. 2005. A practical analytic single scattering model for real time rendering. In *ACM SIGGRAPH 2005 Papers*, 1040–1049.

TOTH, B., AND UMENHOFFER, T. 2009. Real-time volumetric lighting in participating media. *EUROGRAPHICS Short Papers*.

WYMAN, C., AND RAMSEY, S. 2008. Interactive volumetric shadows in participating media with single-scattering. *IEEE Symposium on Interactive Ray Tracing, 2008. RT*.

WYMAN, C. 2011. Voxelized shadow volumes. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ACM, HPG '11, 33–40.

ZHOU, K., HOU, Q., GONG, M., SNYDER, J., GUO, B., AND YEUNG SHUM, H. 2007. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. In *Proc. of Pacific Graphics*.

ZHOU, K., REN, Z., LIN, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Real-time smoke rendering using compensated ray marching. In *ACM SIGGRAPH 2008 papers*, ACM, SIGGRAPH '08, 36:1–36:12.