Technical Report, 8 December 2010
KIT - Karlsruhe Institut of Technology

Instant Multiple Scattering for Interactive Rendering of Heterogeneous Participating Media

Thomas Engelhardt[†], Jan Novak[‡], and Carsten Dachsbacher[§]

Computer Graphics Group
Karlsruhe Institute of Technology
http://cg.ibds.kit.edu

Abstract

Rendering participating media with multiple scattering is costly and often even challenging for off-line methods. In this paper we present a novel method for efficiently rendering such effects that achieves interactive speed for dynamic scenes with both homogeneous and heterogeneous media. It is based on instant radiosity, which is typically used to approximate indirect illumination between surfaces by computing direct lighting from a set of virtual point lights (VPLs). The same principle can be applied to participating media: we describe a particle tracing algorithm to create a set of VPLs within the medium, such that the combined single scattering contribution thereof yields full multiple scattering. Compared to indirect illumination, it is even more important to avoid clamping and singularities from VPLs when rendering participating media effects. For this we derive a GPU-friendly bias compensation for high-quality rendering of participating media with VPLs.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—

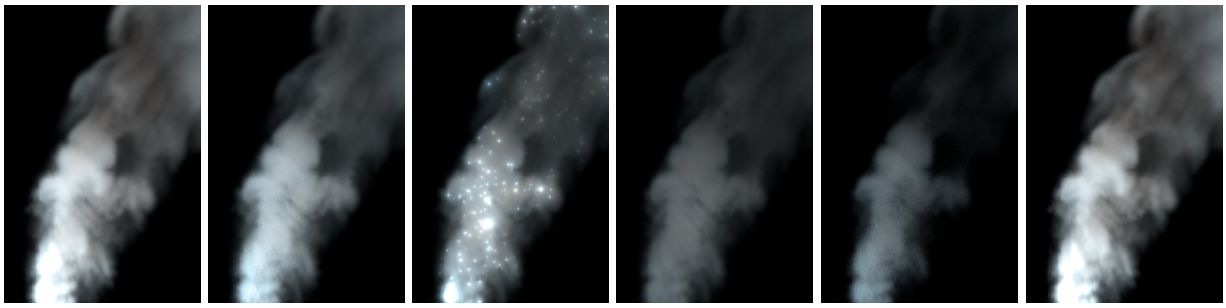
1. Introduction

Global illumination with all its lighting effects, such as shadowing, indirect illumination, caustics, and scattering greatly contributes to the realism of rendered scenes. The computational demands, however, often do not allow a full simulation of light transport in interactive applications. While recent research demonstrated stunning results for indirect illumination [REG*09] and caustics [WWZ*09, YWC*10], the presence of participating media, such as smoke or clouds makes the simulation considerably more difficult. In this case, light does not only interact with surfaces, but is also scattered at potentially every point in space. Various methods exist to compute off-line solutions ranging from conceptually simple methods, e.g. (bidirectional) path tracing [LW93, Vea98], to more elaborate ones such as volumetric photon mapping [JZJ08] and radiance caching [JDZJ07]. In order to achieve interactive speed either preprocessing was necessary [SKSU05, ZRL*08], or simplifying assumptions had to be made so far, e.g. the restriction to single scattering only [ED10, BCRK*ar], or homogeneous media [PSP10].

The goal of our work is to study a GPU-friendly global illumination method for rendering heterogeneous participating media with arbitrary phase functions and multiple scattering. To achieve interactive performance and sufficient flexibility, we build upon instant radiosity [Kel97], which has been widely used in real-time methods for global illumination, however, without, participating media (e.g. see [LSK*07a, RGK*08]). The key idea of instant radiosity (IR) is to break down global illumination into direct lighting from a set of virtual point lights (VPLs). These VPLs are created on the scene's surfaces by using a random walk from the primary light sources. In this spirit, our method also simplifies a complex light transport phenomenon: we create a set of VPLs within the medium such that their *combined single scattering contribution* yields multiple scattering. Note that single scattering in participating media can be efficiently computed using ray marching.

Our method renders full global illumination (GI) including participating media with multiple scattering in a uni-

[†] thomas.engelhardt@kit.edu
[‡] jan.novak@kit.edu
[§] dachsbacher@kit.edu



a) unbiased, 8h50min b) multiple scatt. only c) VPL rendering d) VPLs with clamping e) clamping energy loss f) our method, 5 fps

Figure 1: *(a) Unbiased rendering of smoke using IR and bias compensation (8 hours 50 minutes, CPU). (b) multiple scattering only (8h 49 min, CPU). (c) rendering with VPLs causes the characteristic singularities, which can be avoided by bounding the geometry term (d) (50 min, CPU). (e) note how the energy loss due to bounding is spatially varying in heterogeneous media. Our GPU algorithm can render results with equal high quality as (a) at 5 fps. All images use the same settings for tone mapping.*

fied framework, achieves interactive speed on contemporary GPUs, and requires no preprocessing thus enabling fully dynamic scenes with respect to geometry, phase functions, and heterogeneous media. Akin to IR, our method allows to trade quality for speed by adjusting the number of VPLs, and once the VPLs have been generated, which we can do easily on the CPU, there is no need for any acceleration structure on the GPU to compute GI. However, rendering with IR typically suffers from bright spots when computing the contribution of nearby VPLs due to the unbounded geometry term (Fig. 1c). These can be avoided using clamping, but this introduces bias that becomes visible as energy loss at short distances, which is particularly objectionable in heterogeneous media (Fig. 1d,e). Raab et al. [RSK06] describe a theoretical concept for bias compensation, however, the computation time of their method is prohibitively high (Fig. 1a). Furthermore, it is not naturally amenable to the GPU's rendering pipeline, because it requires tracing arbitrary rays (and thus frequent access to a spatial index structure), and frequent access to all VPLs whenever clamping occurs.

Our main contributions are a particle tracing algorithm for creating VPLs, a scalable bias compensation, and an efficient GPU algorithm for interactive rendering of global illumination including multiple scattering effects. Our compensation allows us to compute results with less accesses to VPLs and faster ray casting thus speeding up rendering. We also describe an approximate version of the compensation for GPU-rendering requiring no ray casting at all.

2. Previous Work

Participating media rendering has been studied extensively in computer graphics. Excellent and comprehensive overviews are given by Pegoraro [Peg09], Cerezo et al. [CPCP*05], and Premože et al. [PARN04]. Early methods for rendering participating media are based on ray tracing [KVH84, RT87], and later on path tracing [PKK00, JDZJ07]. These methods are general, compute unbiased solutions, and can easily handle arbitrary phase functions and heterogeneous media. However, the computational cost is tremendous and thus their application for interactive rendering is beyond question. Interactive photon mapping has been demonstrated on the GPU, e.g. see [WWZ*09, YWC*10], however, participating media rendering is still far from interactive speed [BPPP05, JZJ08], or only at low quality [JMP05]. Zhou et al. [ZRL*08] observe that multiple scattering varies slowly and approximate smoke using RBFs to compute the solution at lower resolution, however, their method requires preprocessing the volume data. Making simplifying assumptions about the participating media, e.g. homogeneous media or isotropic scattering, can lead to fast simulations or even closed form solutions [SGNN04, PSP10]. Highly scattering media can also be modeled using the diffusion approximation [JMLH01], or rendered with light-cuts [AWB08], while single-scattering media can be easily rendered in real-time nowadays using sub-sampling [ED10], hierarchical evaluation [BCRK*ar], or interleaved sampling [TU09].

IR approximates global illumination using direct lighting from a set of VPLs created from random walks. As only relatively few paths (several hundreds) are generated, this method is typically limited to diffuse or slightly glossy materials, see Křivánek et al. [KFB10]. The major cost in IR is the generation of shadow maps for the VPLs; shadow volumes are typically not used, and IR with ray casting is only used in off-line renderers. In recent years several methods have been proposed to speed-up these steps, e.g. by incrementally updating shadow maps [LSK*07a], or using imperfect shadow maps [RGK*08]. Our method also uses VPLs and thus directly benefits from these techniques. Bias compensation for IR has been described for surfaces by Kollig and Keller [KK04] and extended to participating media by Raab et al. [RSK06] (Sect. 4.3). In contrast to our work, their bias compensation targets off-line rendering only and is prohibitively slow. Nevertheless, we build upon their work to derive a practical solution for interactive rendering.

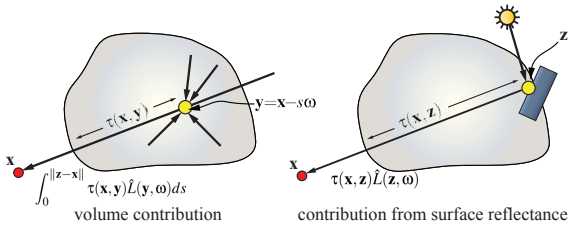


Figure 2: *Light transport in participating media: volume contribution (left) and attenuated surface reflection (right).*

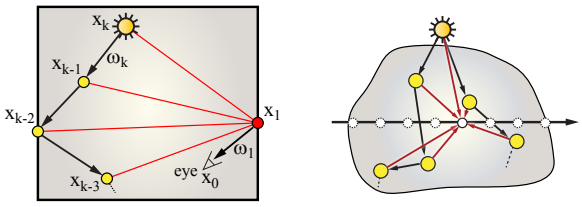


Figure 3: *Left: a light path starting at $\mathbf{x}_k$ is connected to an eye path with a vertex on a surface (red, $\mathbf{x}_1$). Right: participating media require integration along the eye rays, gathering light from precomputed light paths.*

3. Light Transport with Participating Media

In this section we derive the formulation of light transport in scenes with participating media that allows us to create VPLs. The volumetric rendering equation defines the radiance arriving at location $\mathbf{x}$ and going in direction ω as the sum of (1) light emitted and scattered by the volume, and (2) light from surfaces (with $\mathbf{y}=\mathbf{x}-s\omega$; $\mathbf{z}$ is the closest intersection of the ray $\mathbf{x}-s\omega$, $s>0$, see Fig. 2):

$$L(\mathbf{x},\omega) = \underbrace{\int_0^{\|\mathbf{z}-\mathbf{x}\|} \tau(\mathbf{x},\mathbf{y})\hat{L}(\mathbf{y},\omega)ds}_{\text{volume contribution}} + \underbrace{\tau(\mathbf{x},\mathbf{z})\hat{L}(\mathbf{z},\omega)}_{\text{surface reflection}}. \quad (1)$$

The transmittance $\tau(\mathbf{x},\mathbf{y}) = e^{\left(\int_0^{\|\mathbf{x}-\mathbf{y}\|} \sigma_t(x-t\omega)dt\right)}$ accounts for out-scattering and absorption along a ray segment; the extinction coefficient $\sigma_t(\mathbf{x})$ is the sum of the coefficients for absorption, $\sigma_a(\mathbf{x})$, and scattering, $\sigma_s(\mathbf{x})$. Radiance integrated along the ray and light reflected from the surface can be expressed using the general light transport equation:

$$\hat{L}(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} \hat{f}(\mathbf{x},\omega,\omega')\hat{L}(\mathbf{x},\omega')d\omega'.$$

$\hat{f}(\mathbf{x},\omega,\omega')$ defines scattering using the phase function, f_p, in the volume, and the BRDF, f_r, on the surfaces as:

$$\hat{f}(\mathbf{x},\omega,\omega') = \begin{cases} f_p(\mathbf{x},\omega,\omega')\sigma_s(\mathbf{x}) & \text{if } \mathbf{x} \text{ is in the medium} \\ f_r(\mathbf{x},\omega,-\omega')\cos^+(\theta) & \text{if } \mathbf{x} \text{ lies on a surface} \end{cases}$$

where θ is the angle between the normal at $\mathbf{x}$ and $-\omega'$. Each of the distribution functions describes the probability of light being scattered from direction ω' to direction ω [PH10].

Recursively expanding Eq. 1 yields the light incident at a point $\mathbf{x}$ for all paths of length up to n [PKK00]:

$$L^n(\mathbf{x}_0,\omega_1) = \sum_{k=1}^{n} \underbrace{\int..\int}_{\text{k-times}} L_e(\mathbf{x}_k,\omega_k)\mathbf{T}_{1,k}\tau(\mathbf{x}_1,\mathbf{x}_0)d\mu(\mathbf{x}_1)..d\mu(\mathbf{x}_k)$$

where ω_k is the direction from $\mathbf{x}_k$ to $\mathbf{x}_{k-1}$ (Fig. 3), and L^∞ represents the full light transport in a scene. $\mathbf{T}_{j,k}$ is the path throughput defined using the *generalized geometry term* $\hat{G}(\mathbf{x},\mathbf{y})$ and *visibility term* $\hat{V}(\mathbf{x},\mathbf{y})$ as:

$$\mathbf{T}_{j,k} = \prod_{i=j}^{k-1} \hat{f}(\mathbf{x}_i,\omega_i,\omega_{i+1})\hat{G}(\mathbf{x}_i,\mathbf{x}_{i+1})\hat{V}(\mathbf{x}_i,\mathbf{x}_{i+1}), \text{ with}$$

$$\hat{G}(\mathbf{x},\mathbf{y}) = D_\mathbf{x}(\mathbf{y})D_\mathbf{y}(\mathbf{x})/\|\mathbf{x}-\mathbf{y}\|^2 \text{ and}$$

$$\hat{V}(\mathbf{x},\mathbf{y}) = \tau(\mathbf{x},\mathbf{y})V(\mathbf{x},\mathbf{y}),$$

where $V(\mathbf{x},\mathbf{y})$ is the binary visibility function, $D_\mathbf{x}(\mathbf{y}) = 1$ and $d\mu(\mathbf{x}) = dV(\mathbf{x})$ in the case when $\mathbf{x}$ is in the medium. And otherwise, when light is reflected at a surface location $\mathbf{x}$, $D_\mathbf{x}(\mathbf{y}) = \max(0,\mathbf{n}_\mathbf{x}\cdot\omega_{xy})$ and $d\mu(\mathbf{x}) = dA(\mathbf{x})$. To derive our rendering algorithm we reformulate the path integral:

$$L^\infty(\mathbf{x}_0,\omega_1) = L^2(\mathbf{x}_0,\omega_1) + \sum_{k=3}^{\infty} \underbrace{\int..\int L_e(\mathbf{x}_k,\omega_k)\mathbf{T}_{2,k}}_{\text{represented as VPLs}} \mathbf{T}_{1,2}\tau(\mathbf{x}_1,\mathbf{x}_0)d\mu(\mathbf{x}_1)..d\mu(\mathbf{x}_k). \quad (2)$$

Here, we first split the *path space* into short paths of length less or equal to 2, and long paths. The short paths account for light emitted from primary light sources, eventually followed by one reflection or scattering event. They can be efficiently computed, e.g. using shadow mapping and ray marching; the original IR approach renders direct light with VPLs as well. The important observation is, that we can split *long paths* as well and separate the last two path segments, which again can be computed easily. The path prefix $(\mathbf{x}_k,\mathbf{x}_{k-1},..,\mathbf{x}_2)$ can be precomputed using a random walk and stored as VPLs. Connecting the eye path segment $(\mathbf{x}_1,\mathbf{x}_0)$ to the VPLs then approximates indirect illumination.

4. Instant Multiple Scattering

Next we describe the individual steps of our method. As with instant radiosity, we start with the VPL generation (Sect. 4.1), followed by the rendering procedure (Sect. 4.2). Lastly we introduce our bias compensation and discuss different variants regarding a GPU-friendly implementation.

4.1. Generation of Virtual Point Lights

Using the random walk we incrementally construct N paths (yielding M VPLs) by importance sampling the Monte Carlo estimator for the path integral. Note that in contrast to IR with surfaces only, we do not only sample directions (at

light-surface interactions), but also distances along rays as light might be scattered when traveling through the medium. Raab et al. [RSK06] describe the principle of this process, however, as this is an integral part of our method we detail the steps in the following. We can derive the Monte Carlo estimator by adapting it from Veach [Vea98]:

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{k=1}^{\infty}\frac{L_e(\mathbf{x}_k,\omega_k)}{p(\mathbf{x}_k,\omega_k)}\frac{\tau(\mathbf{x}_k,\mathbf{x}_{k-1})}{p(t_k)}\prod_{j=1}^{k-1}\frac{\tau(\mathbf{x}_j,\mathbf{x}_{j-1})\hat{C}_j}{p(\omega_j)p(t_j)},$$

$$\hat{C}_j=\begin{cases} f_p(\mathbf{x}_j,\omega_j,\omega_{j+1}) & \text{if } \mathbf{x}_j \text{ is in the medium}\\ f_r(\mathbf{x}_j,\omega_j,-\omega_{j+1})\mathbf{n}_{\mathbf{x}_j}\cdot\omega_j & \text{if } \mathbf{x}_j \text{ is on a surface}\end{cases}$$

where $p(.)$ is the probability density function (PDF) for sampling a position $\mathbf{x}_k$ on a light source, a direction ω, or distance t along a ray, respectively.

Note that strictly speaking, we generate paths and store path vertices, whereas IR creates point lights. The reason is that we support arbitrary BRDFs and phase functions and rather store them together with the path vertices instead of turning them into an intensity distribution for VPLs. Nevertheless we will use the term VPL tantamount with path vertices in the style of IR. However, in contrast to IR, we do not simply output a set of VPLs, but also keep track which VPLs belong to the same random walk. We will later use this information to accelerate the bias compensation (Sect. 4.4).

Akin to IR [Kel97], our random walk begins by creating a first path vertex on a primary light source transporting the radiance $L(\mathbf{x}_k,\omega_k)=L_e(\mathbf{x}_k,\omega_k)/p(\mathbf{x}_k,\omega_k)$ (ω_k is sampled according to the light source emission), and we incrementally construct the path with:
1) Sample next scattering event with direction ω_j (according to $\hat{f}(\mathbf{x}_j,..)$ if the current vertex is not on the light source, i.e. $j<k$), and distance t_j along the ray $\mathbf{r}(t)=\mathbf{x}_j+t\omega_j$ to determine the next path vertex $\mathbf{x}_{j-1}$. It can either reside in the volume, or on the closest surface intersected by the ray (which would always be the case in vacuum). Scattering in participating media is a stochastic process and thus an interaction might occur at a shorter distance. For homogeneous media analytical PDFs exist [LLL*96], for heterogeneous media we either resort to ray marching [LLL*96] or sample according to maximum extinction [RSK06]. The path's incident radiance at the interaction location is then:
$L(\mathbf{x}_{j-1},\omega_j)=L(\mathbf{x}_j,\omega_j)\tau(\mathbf{x}_j,\mathbf{x}_{j-1})\hat{C}_j/p(\omega_j,t_j)$.
2) Create a VPL at $\mathbf{x}_{j-1}$ storing the incident radiance $L(\mathbf{x}_{j-1},\omega_j)$ and incident direction ω_j. If the VPL is created in the volume we store the scattering coefficient $\sigma_s(\mathbf{x}_{j-1})$ and phase function with the VPL, otherwise the BRDF and surface normal at $\mathbf{x}_{j-1}$ (or tangent space for anisotropic BRDFs), respectively.
3) Possibly terminate the path with Russian roulette with a probability of $1-\mathbf{T}_{j-1,k}$. If the path is continued, we proceed with step 1, otherwise we store the VPLs generated in this random walk together.

4.2. Rendering Participating Media with VPLs

The VPLs generated by the random walk are then used to compute the lighting in the scene. We separate this process into lighting of surfaces and volume contribution (Eq. 1).

Surface Lighting The indirect illumination on surfaces is computed from all VPLs no matter if they reside on surfaces or in the volume. Let $\mathbf{v}_i$ and ω_i be the location and direction of the i-th VPL (with radiance $L_{V,i}$) to $\mathbf{x}_1$, then the indirect light is (M VPLs, N paths):

$$L_s(\mathbf{x}_0,\omega_1)\approx\frac{\tau(x_0,\mathbf{x}_1)}{N}\sum_{i=1}^{M}f_r(\mathbf{x}_1,\omega_1,\omega_i)\tau(\mathbf{x}_1,\mathbf{v}_i)\hat{G}(\mathbf{x}_1,\mathbf{v}_i)L_{V,i}.$$

For surface lighting, the general scattering function can be replaced by the BRDF f_r. The generalized geometry term $\hat{G}$, however, remains, as we also gather light from VPLs in the volume. We compute the visibility function in $\hat{V}$ using shadow mapping (see Sect. 5). The radiance from a VPL is computed by evaluating the stored BRDF or the combination of phase function and scattering coefficient, respectively.

Volume Contribution To evaluate the multiple scattered light, we have to integrate the single scattering contribution from all VPLs along an eye ray using ray marching. However, the evaluation of the inscattered light at every step along the ray is very costly, in particular due to the large number of VPLs and the transmittance computation (in heterogeneous media) from the ray to the VPL. We found that interleaved sampling of *VPL paths* often yields hardly distinguishable results from the full evaluation (for reasonable many VPLs): we use R stratified samples, $\mathbf{r}_i$, along the ray, and at every sample compute the inscattering from a randomly chosen subset of random walks only. Note that this introduces no bias, and is not noticeable in practice as we generate 128 to 1024 random walks (resulting in up to 4000 VPLs total). Thus the volume contribution (inscattering only, no emission) is computed as:

$$L_v(\mathbf{x}_0,\omega_1)\approx\sum_{i=1}^{K}\frac{\tau(\mathbf{x}_0,\mathbf{r}_i)}{N}\sum_{j\in P_i}p(\mathbf{r}_i,\omega_1,\omega_{i,j})\tau(\mathbf{r}_i,\mathbf{x}_j)\hat{G}(\mathbf{r}_i,\mathbf{v}_j)L_{V,j},$$

where the second summation iterates over the VPLs of the randomly chosen subset of paths, P_i, and $\omega_{i,j}$ is the direction from the j-th VPL to $\mathbf{r}_i$. The cost for L_v is roughly equal to that of L_s, if P_i contains only one path.

4.3. Bias Compensation

Lighting as described in Sect. 4.2 leads to unbiased solutions, but suffers from bright spots in the rendering due to the singularities in the geometry term. Often clamping the geometry term $\hat{G}'(\mathbf{x},\mathbf{y})=min(\hat{G}(\mathbf{x},\mathbf{y}),b)$ using some bound $b>0$ is used to remove the high intensity peaks; this, however, introduces bias (please see [KK04, RSK06] for a discussion how to choose b). Kollig et al. [KK04] recursively correct the bias for inter-surface light transport and Raab et al. [RSK06] extend this idea to participating media. Both

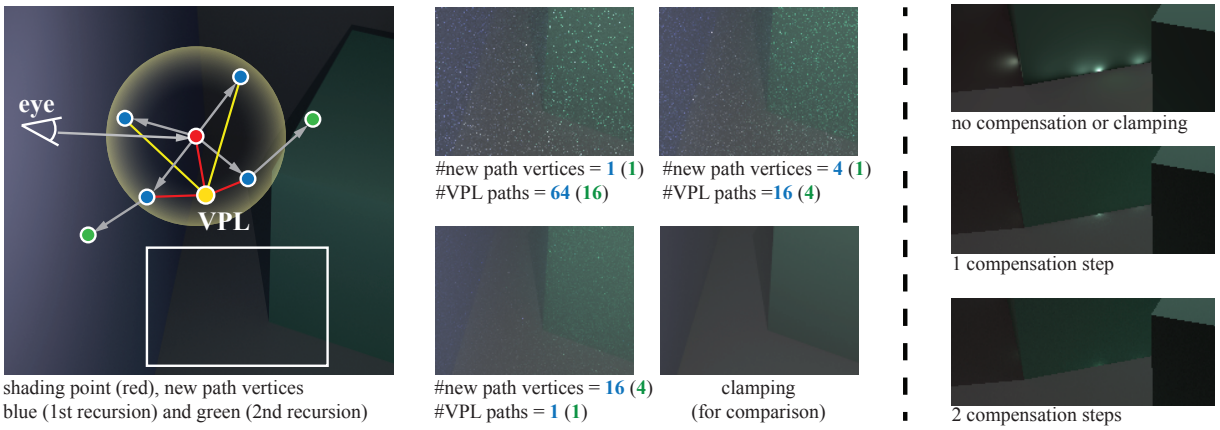


Figure 4: *Left: compensation for the shading point (red) is necessary and new path vertices are generated. Two of which (blue) require a second compensation resulting in the green path vertices. Center: rendering with different sub-sampling strategies (only close-ups shown, approx. same computation time). Let V_i be the number of new path vertices of the i-th compensation step, and P_i the number of VPL paths to which they are connected. Then the total number of paths with recursion depth 2 is $V_1 \cdot (P_1 + V_2 \cdot P_2)$ (320 is all three cases). Note that clamping results in much darker images. Right: singularities from lighting with VPLs (top), and using unclamped VPLs after one (middle) and two (botton) compensation steps.*

compute a biased solution with clamping first, and afterwards compute a correction term, $L'(\mathbf{x},\omega)$, for every shading location where clamping occured:

$$L'(\mathbf{x},\omega) = \int_{S^2} L(\mathbf{x},\omega') \frac{\max(0,\hat{G}(\mathbf{x},\mathbf{x}')-b)}{\hat{G}(\mathbf{x},\mathbf{x}')} \hat{f}(\mathbf{x},\omega,\omega')\tau(\mathbf{x},\mathbf{x}')d\omega'.$$

The correction works as follows (Fig. 4): from the shading location $\mathbf{x}$, a ray is cast into a direction chosen according to $\hat{f}(.)$. Along this ray either scattering in the medium occurs, or the ray intersects another surface (as in Sect. 4.1). The location of this interaction, $\mathbf{x}'$, becomes a vertex of a new path and is connected to all VPLs. The contribution of the new path is weighted by $\max(0,(\hat{G}(\mathbf{x},\mathbf{x}')-b)/\hat{G}(\mathbf{x},\mathbf{x}'))$ [KK04, RSK06]. This weight becomes zero when $\mathbf{x}'$ is too far from $\mathbf{x}$ (outside the region where the clamping occurs), i.e. the contribution of this path has already been accounted for by the VPLs. This, however, means that some new paths are generated superfluously as they do not contribute to the compensation. Note that clamping of a VPL's contribution might also occur at the new vertex, $\mathbf{x}'$, which needs to be corrected recursively in the same way. It is well-known that the contribution of the compensation drops exponentially with the recursion depth. We will exploit this fact and discuss how the recursion can be terminated introducing no (visible) bias.

4.4. Accelerated Bias Compensation

Unfortunately the bias compensation is very costly as it requires ray casting to find new vertices, all VPLs have to be accessed at every new path vertex, and it can degenerate to (bidirectional) path tracing [DKH*ar]. In particular in dense media, it is to be expected that clamping is necessary at almost every location in space. In the following we present our observations that allow for several optimizations. As mentioned before, our utmost goal is to remove *visible bias*, eventually not producing the exact mathematical result; we discuss this along with the proposed simplifications.

Sub-Sampling Random Walks For unbiased results it is not necessary to connect a new vertex to all VPLs. Instead, we can just connect to a randomly chosen subset of *light paths*. We opt for this, because randomly choosing VPLs instead of entire paths yields wrong results if they are not correctly reweighted, which in turn requires information on the respective paths again. Sub-sampling the paths allows us to adjust the sampling parameters for the compensation more freely, e.g. we can create more new path vertices and connect to a smaller set of VPL-paths at the same cost (Fig. 4). According to our experiments, creating more new path vertices is more important than connecting each individual vertex to all light paths. This particularly holds when $\hat{G}(\mathbf{x},\mathbf{x}')$ varies strongly within the bounding region, e.g. in corners as shown in Fig. 4, or with strongly varying transmittance $\tau(\mathbf{x},\mathbf{x}')$ or scattering functions at $\mathbf{x}$.

Generation of New Vertices Since the contribution of the new paths is zero whenever they are constructed using vertices outside the clamping region, we can speed up the computation by restricting the new vertices to lie within that region. To sample a new distance, we assume the participating media to be *locally homogeneous*. This means that we use the average extinction coefficient, $\overline{\sigma_t}(\mathbf{x})$, in the proximity of $\mathbf{x}$ for sampling the distance. This value can be efficiently obtained from a downsampled representation of the medium, e.g. if it is stored in a 3D texture on the GPU. The probability density function for sampling within the radius, d, of the clamping region then becomes $p(t) = (\sigma_t \exp(-\sigma_t t))/(1-$

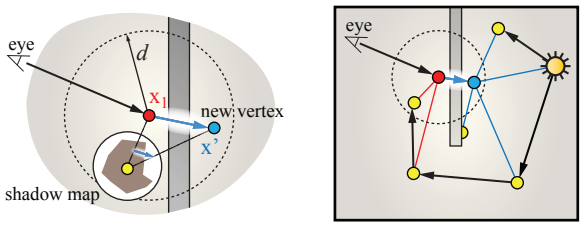


Figure 5: *Left: we use shadows maps of nearby VPLs to approximate ray surface intersections. Right: omitting visibility between* $\mathbf{x}_1$ *and* $\mathbf{x}'$ *can theoretically lead to light leaking.*

$\exp(-\sigma_t d)$). Using this PDF we compute the new distance as $t = -\ln(1-\xi\alpha)/\sigma_t$ with $\alpha = (1-\exp(-\sigma_t d))$ using a random number $\xi \in [0;1)$. Note that if there is an intersection of the random ray closer to $\mathbf{x}$ than to $\mathbf{x}+t\cdot\omega$, then this intersection becomes the new path vertex.

For heterogeneous media we can sample distance as described by Lafortune et al. [LLL*96]. However, we found that assuming locally homogeneous media again simplifies sampling without noticeable impact on performance or quality. Note that this only affects sampling, whereas the result remains unbiased as long as we correctly compute the transmittance $\tau(\mathbf{x},\mathbf{x}')$. We can take this assumption one step further and also approximate the transmittance using $\overline{\sigma_t}(\mathbf{x})$. In all our test scenes this yields indistinguishable results from ground truth, although the assumption obviously fails at locations with strongly varying extinction. Note that these locations could be easily detected using the gradient of $\sigma_t(\mathbf{x})$.

Local Visibility Besides VPL lighting and creating new paths, using ray casting to determine whether a new path segment intersects a surface before scattering occurs contributes most to the cost of bias compensation. VPL-based rendering typically uses shadow mapping to resolve visibility in the geometry term. Thus visibility in the scene has already been evaluated at a large number of locations, namely those of the VPLs. Obviously it is beneficial to reuse this data, in particular for GPU-implementations where ray casting would require elaborate spatial index structures. Clamping occurs when a VPL and a shading location are close, and in this case the VPL's shadow map provides a good (sampled) representation of nearby surfaces. We can then ray cast using the shadow map as geometry representation as described by [YWC*10]. Note that additional surface data akin to a reflective shadow map [DS05] is necessary to evaluate the entire light paths. We can further optimize the intersection test and use the minimum depth value in a VPL's shadow map (e.g. computed efficiently using parallel reduction) to conservatively test if there are surfaces close enough to be intersected, and omit ray casting if this is not the case. A more rigorous simplification is to totally omit visibility for the path vertex generation, i.e. to always use $\mathbf{x}' = \mathbf{x}+t\cdot\omega$. This correctly removes bias in volumes, and only can cause errors close to surfaces. Fig. 5 illustrates when errors would occur, however, we were not able to produce any scene where this error would have been noticeable. This is because either $\mathbf{x}$ and $\mathbf{x}'$ are nearby and it is less likely that occlusion might occur inbetween, or they are further apart and the quadratically decreasing compensation term outweighs.

Limiting Recursion Depth Clamping of a VPL's contribution might also occur at new path vertices, which in turn triggers another bias compensation step. However, the compensation integral convolves the gathered radiance with the generalized scattering function, $\hat{f}$, having two consequences: first, the convolution removes high frequencies (see Sect. 7 for a discussion on specular surfaces and anisotropy). We can exploit this and still compute unbiased results when stopping the compensation using the unclamped contribution from the VPLs. Using this for recursion depth 2 or higher yielded only hardly visible differences in our test scenes (Fig.4). Second, the amount of compensation drops exponentially with the recursion depth. For our GPU-implementation, where recursion is very costly, we compensate once without recursion and connect to VPLs with reduced clamping, just enough to not produce bright spots. This largely compensates bias, but of course not completely.

5. Implementation Details

We implemented our method using Direct3D 11 and also integrated it into a custom offline renderer for evaluation purposes. Random walks and their throughput are always computed on the CPU using a bounding volume hierarchy (BVH), with surface area heuristics, as acceleration structure; for dynamic objects the BVH is recomputed every frame and merged into the BVH of the static scene. The cost for computing the random walks was negligible, even for complex scenes. We use a Mersenne Twister as pseudo-random number generator and for each random walk we use the same random seed for all frames to maintain coherency.

As the CPU implementation is straightforward, we restrict ourselves to the peculiarities of the GPU implementation. Rendering on the GPU is split into 4 components: direct and indirect illumination, and single and multiple scattering. Indirect illumination and multiple scattering are initially computed at lower (1/64) resolution and refined at pixels where bilateral upsampling does not yield satisfactory results, as described in [REG*09]. The transmittance is computed analytically for homogeneous media, and using ray marching through a 3D texture, storing scattering and absorption coefficients, for heterogeneous media; we use the Henyey-Greenstein approximation for varying phase functions. For shadow mapping we use point-based rendering akin to imperfect shadow maps [RGK*08], however, combined with the octahedron parameterization [ED08] to not waste texture space; for each VPL we create a 128^2 texture map.

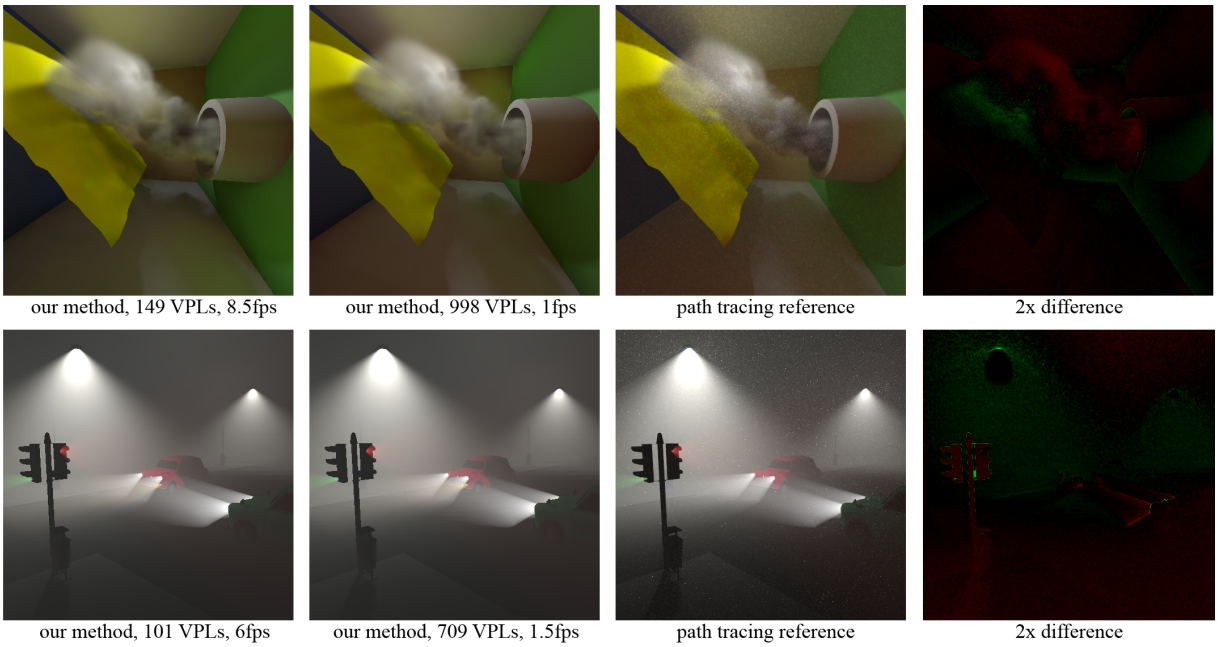


Figure 7: *Two test scenes rendered using our method with bias compensation (1 recursion), local visibility approximation, and different numbers of VPLs. The images on the right show the difference of the 2nd column's images to path tracing scaled by 2 (green means too dark, red too bright). Please see the accompanying video for more examples.*

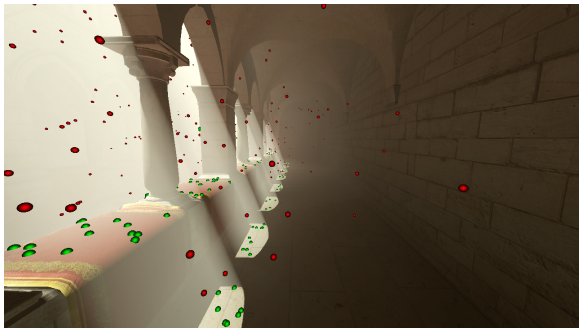


Figure 6: *The Crytek Sponza scene (262k triangles) rendering at 8 fps using approx. 1290 VPLs. VPLs residing on surfaces are shown as green, VPLs in the volume as red balls.*

6. Results

We evaluated our method using several test scenes with homogeneous and heterogeneous participating media. All timings have been recorded using an Intel Core i7 system at 2.8GHz with an ATI Radeon 5870 GPU.

Fig. 6 shows the Crytek sponza model with a homogeneous highly scattering medium. It has been rendered at about 8 frames per second with multiple scattering and our GPU-friendly bias compensation and local visibilty approximation. Although the volume is homogeneous we use many VPLs (1290 on average) because of the spatial extend of the scene. Fig. 7 shows a comparison of our method with different quality settings to path tracing reference images. In both scenes high quality results are achieved with relatively few VPLs, as the medium is either homogenous or spatially bounded. The transmittance in scenes with homogeneous medium has been computed analytically, and using ray marching otherwise. The smoke scene is rendered with 500 steps for ray marching along the eye ray, and 64 steps for computing the transmittance when gathering light from VPLs; the car scene uses 200 steps for single scattering.

7. Analysis and Discussion

In this section we describe findings from experimenting with our method, which we believe are important to assess its strengths and limitations.

Similar to classic IR, our method sub-samples the path space. Despite this sub-sampling, the result images are close to ground truth, the reason being that single scattering and transmittance are computed at high precision using ray marching, which preserves crisp features. This can be compared to applying textures to modulate the indirect lighting in IR. The amount of "global" features that are captured depends on the number of VPLs. Consequently, more VPLs are required for dense and heterogeneous than for thin or homogeneous media. However, in contrast to surface lighting, ray marching accounts for a significant part of the computation time, and the cost for VPL generation and shadow mapping is less critical.

Our method supports anisotropic phase functions, but

strong backward and forward scattering causes problems similar to glossy materials with IR [KFB10] (please see the video for an example scene). This is simply because subsampling the path space assumes smooth illumination.

We also applied incremental IR [LSK*07b] to our method. As long as the scene geometry is static and only the media changes, we can keep all paths and only recompute the path throughputs, or optionally update some paths per frame only. This saves costly shadow map computation which amounts to approx. 60ms per 1000 VPLs in our example scenes.

8. Conclusions and Future Work

We presented a novel method for rendering global illumination including multiple scattering in heterogeneous media, which is based on instant radiosity thus requiring no precomputation. Using our scalable bias compensation technique and making use of GPU-friendly data structures it achieves interactive speed on contemporary graphics hardware.

Our method is a first step bringing instant radiosity with participating media to interactive applications. As such, it lends itself as a basis for studying improvements that have been developed for the classic instant radiosity method, e.g. bidirectional generation of VPLs.

References

[AWB08] ARBREE A., WALTER B., BALA K.: Single-pass scalable subsurface rendering with lightcuts. *Computer Graphics Forum 27*, 2 (2008), 507–516.

[BCRK*ar] BARAN I., CHEN J., RAGAN-KELLEY J., DURAND F., LEHTINEN J.: A hierarchical volumetric shadow algorithm for single scattering. In *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2010, to appear).

[BPPP05] BOUDET A., PITOT P., PRATMARTY D., PAULIN M.: Photon splatting for participating media. In *Proc. of GRAPHITE '05* (2005), pp. 197–204.

[CPCP*05] CEREZO E., PEREZ-CAZORLA F., PUEYO X., SERON F., SILLION F.: A survey on participating media rendering techniques, 2005.

[DKH*ar] DAVIDOVIC T., KRIVANEK J., HASAN M., SLUSALLEK P., BALA K.: Combining global and local lights for high-rank illumination effects. In *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2010, to appear).

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective Shadow Maps. In *Proc. of I3D* (2005), pp. 203–213.

[ED08] ENGELHARDT T., DACHSBACHER C.: Octahedron environment maps. In *Proc. of Vision, Modelling and Visualization* (2008), pp. 383–388.

[ED10] ENGELHARDT T., DACHSBACHER C.: Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Proc. of I3D* (2010), pp. 119–125.

[JDZJ07] JAROSZ W., DONNER C., ZWICKER M., JENSEN H. W.: Radiance caching for participating media. In *SIGGRAPH '07: ACM SIGGRAPH 2007 sketches* (2007), p. 56.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *SIGGRAPH '01* (2001), pp. 511–518.

[JMP05] JIMÉNEZ J.-R., MYSZKOWSKI K., PUEYO X.: Interactive global illumination in dynamic participating media using selective photon tracing. In *Proc. of SCCG* (2005), pp. 211–218.

[JZJ08] JAROSZ W., ZWICKER M., JENSEN H. W.: The beam radiance estimate for volumetric photon mapping. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes* (2008), pp. 1–112.

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97* (1997), pp. 49–56.

[KFB10] KŘIVÁNEK J., FERWERDA J. A., BALA K.: Effects of global illumination approximations on material appearance. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 29*, 4 (2010), 1–10.

[KK04] KOLLIG T., KELLER A.: Illumination in the presence of weak singularities. *Monte Carlo and Quasi-Monte Carlo methods* (2004).

[KVH84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. *Computer Graphics (Proc. of SIGGRAPH) 18*, 3 (1984), 165–174.

[LLL*96] LAFORTUNE E. P., LAFORTUNE E. P., LAFORTUNE E. P., WILLEMS Y. D., WILLEMS Y. D.: Rendering participating media with bidirectional path tracing. In *Proc. of the Eurographics Rendering Workshop* (1996), pp. 91–100.

[LSK*07a] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. of the Eurographics Symposium on Rendering* (2007), pp. 277–286.

[LSK*07b] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. of Eurographics Symposium on Rendering* (2007), pp. 277–286.

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *Proc. of Computegraphics '93* (1993), pp. 145–153.

[PARN04] PREMOŽE S., ASHIKHMON M., RAMAMOORTHI R., NAYAR S.: Practical rendering of multiple scattering effects in participating media. In *Proc. of Eurographics Symposium on Rendering* (2004).

[Peg09] PEGORARO V.: *Efficient Physically-Based Simulation of Light Transport in Participating Media*. School of Computing, University of Utah, Ph.D. Thesis, 2009.

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2010.

[PKK00] PAULY M., KOLLIG T., KELLER A.: Metropolis light transport for participating media. In *Proc. of the Eurographics Workshop on Rendering Techniques* (2000), pp. 11–22.

[PSP10] PEGORARO V., SCHOTT M., PARKER S. G.: A Closed-Form Solution to Single Scattering for General Phase Functions and Light Distributions. *Computer Graphics Forum (Proc. EGSR) 29*, 4 (2010), 1365–1374.

[REG*09] RITSCHEL T., ENGELHARDT T., GROSCH T., SEIDEL H.-P., KAUTZ J., DACHSBACHER C.: Micro-rendering for scalable, parallel final gathering. In *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2009), pp. 1–8.

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. In *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2008), pp. 129:1–129:8.

[RSK06] RAAB M., SEIBERT D., KELLER A.: Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods* (2006), pp. 591–606.

[RT87] RUSHMEIER H. E., TORRANCE K. E.: The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics (Proc. of SIGGRAPH) 21*, 4 (1987), 293–302.

[SGNN04] SRINIVASA G. NARASIMHAN R. R., NAYAR S. K.: Analytic rendering of multiple scattering in participating media. Columbia University Technical Report, 2004.

[SKSU05] SZIRMAY-KALOS L., SBERT M., UMENHOFFER T.: Real-time multiple scattering in participating media with illumination networks. *Proc. of the Eurographics Symposium on Rendering* (2005), 1315–1324.

[TU09] TÓTH B., UMENHOFFER T.: real-time volumetric lighting in participating media. In *Eurographics Short Papers* (2009).

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, 1998.

[WWZ*09] WANG R., WANG R., ZHOU K., PAN M., BAO H.: An efficient gpu-based approach for interactive global illumination. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)* (2009), pp. 1–8.

[YWC*10] YAO C., WANG B., CHAN B., YONG J., PAUL J.-C.: Multi-image based photon tracing for interactive global illumination of dynamic scenes. *Computer Graphics Forum (Proc. EGSR) 29*, 4 (2010), 1315–1324.

[ZRL*08] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.-Y.: Real-time smoke rendering using compensated ray marching. In *ACM Transactions on Graphics (Proc. of SIGGRAPH)* (2008), pp. 1–12.