Interactive Cloud Rendering Using Temporally-Coherent Photon Mapping

Oskar Elek[a], Tobias Ritschel[a], Alexander Wilkie[b], Hans-Peter Seidel[a]

[a]*MPI Informatik, Germany*
[b]*Charles University, Czech Republic*

Abstract

This work presents a novel interactive algorithm for simulation of light transport in clouds. Exploiting the high temporal coherence of the typical illumination and morphology of clouds we build on volumetric photon mapping, which we modify to allow for interactive rendering speeds – instead of building a fresh irregular photon map for every scene state change we accumulate photon contributions in a regular grid structure. This is then continuously being refreshed by re-shooting only a fraction of the total amount of photons in each frame. To maintain its temporal coherence and low variance, a low-resolution grid is initially used, and is then upsampled to the density field resolution on a physical basis in each frame. We also present a technique to store and reconstruct the angular illumination information by exploiting properties of the standard Henyey-Greenstein function, namely its ability to express anisotropic angular distributions with a single dominating direction. The presented method is physically-plausible, conceptually simple and comparatively easy to implement. Moreover, it operates only above the cloud density field, thus not requiring any precomputation, and handles all light sources typical for the given environment, i. e., where one of the light sources dominates.

Keywords: participating media; natural phenomena; real-time rendering; photon mapping; physically-based rendering
2000 MSC: 68U05, 65D18

1. Introduction

Radiative transport in clouds and participating media in general is an important and broad problem. Specifically, interactive cloud rendering is often needed when an application includes interactive visualization of outdoor environments. That includes serious applications such as flight and soaring simulators or meteorological visualizations, but also popular applications, for instance 3D games and packages like Google Earth.

Among participating media, clouds stand out as a particularly difficult case for state-of-the-art volumetric light transport techniques. Their virtually unit albedo and small relative mean free photon path imply simulation of many scattering orders. The very high scattering anisotropy of cloud droplets in turn causes difficulties for purely ray-based methods [37] and approaches based on discrete light propagation volumes [7, 14, 12]. The latter property in combination with the usual high complexity of cloud shapes also makes diffusion-based approaches [40, 32] unsuitable, mainly because it is usually difficult to fulfil the boundary conditions of the diffusion equation. Finally, the high variability of morphological cloud types makes it difficult to establish an efficient representation of their mass distributions. Naturally, these issues are even more challenging for interactive rendering techniques.

The method presented in this work contributes to the topic in question in these main ways:

- We present a general scheme for propagation of light energy inside participating media using photon beams. The energy is stored in a regular grid-like photon map, which is decoupled from the cloud density field. This allows a continuous updating of the photon map by re-shooting only a fraction of the photons that constitute the scene illumination in each frame. Although this potentially causes the photon map to contain a partially obsolete illumination solution, in slowly-changing environments such as clouds the difference from the ideal solution will typically be negligible.
- To avoid the necessity of explicitly storing individual photons we introduce a specialized representation of the angular illumination information. This representation accumulates the average cosine of the stored photons in respect to a reference direction (usually the direction of the strongest light source present in the scene). This accumulated value in fact is the asymmetry factor of the angular photon distribution in each cell, and is subsequently used to reconstruct the directionally-dependent illumination by evaluating the Henyey-Greenstein function parametrized by this factor and by the reference direction.
- Additionally we present an efficient physically-based upsampling scheme to improve the resolution of the photon map by using an additional knowledge available to us in the simulation. The upsampling is a fast parallel operation and allows to dramatically decrease the resolution of the photon map, along with the number of traced photons required for obtaining a noise-free solution.
- The presented method natively maps well to modern GPUs, which we show by our own implementation. Despite not containing any specific low-level optimizations it is capable of simulating the radiative transport process in clouds at interactive speeds.

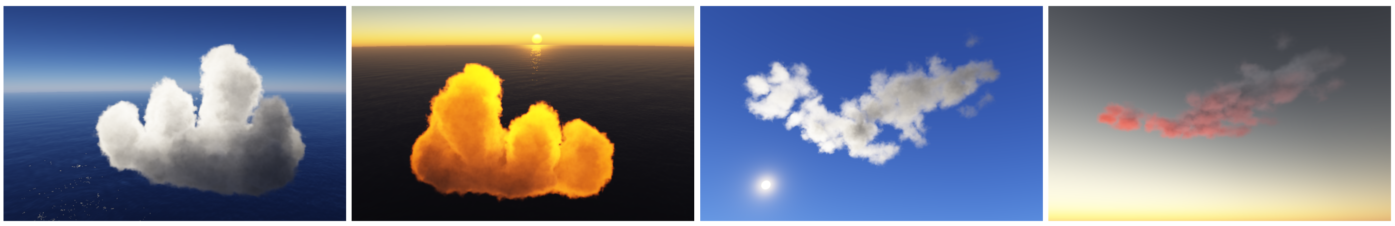

Figure 1: Cumulus congestus and Cirrocumulus altocumulus rendered fully dynamically by our approach at 50 and 90 FPS, respectively.

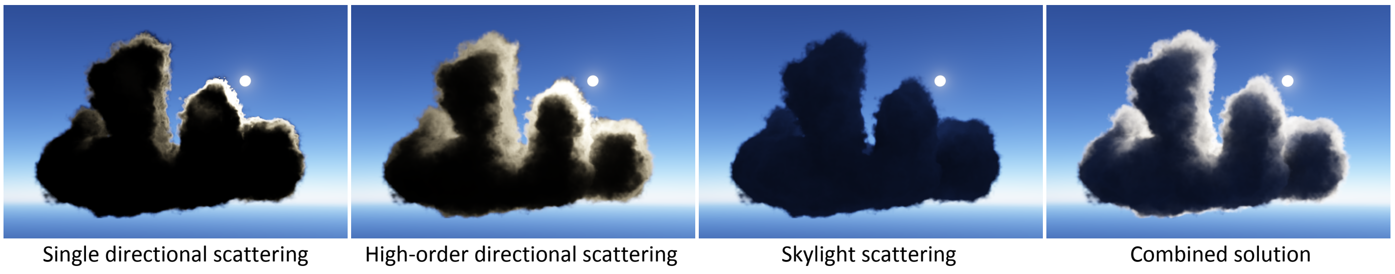
Single directional scattering
High-order directional scattering
Skylight scattering
Combined solution

Figure 2: Decomposition of the three basic scattering categories in clouds. Note that the transition from single to second and higher orders of scattering is very subtle due to the high scattering anisotropy. For this reason methods that handle single and multiple scattering separately necessarily produce inconsistent results.

2. Related Work

A large volume of research concerns itself with participating media and related subjects [6]. We will therefore limit ourselves to works directly relevant in the current context.

Photon mapping. The intrinsic ability of the basic photon mapping algorithm to cache radiant energy in the scene shows to be very beneficial for participating media rendering. This application has first been described by Jensen and Christensen [21]. Later, Jarosz et al. [20] developed a new beam gathering technique specifically for participating media and subsequently generalized it also for propagation of the radiant energy itself [19]. Their contribution represents a significant improvement over the classical point-based photon mapping, although it remains to be seen how it would cope with the very high amount of scattering orders encountered in clouds.

Several attempts to speed-up photon mapping by continuously updating the photon map exist. Airieau et al. [1] developed an interactive technique for continuous streaming of photons, where parts of the scene where illumination changes more rapidly are updated with higher priority. A similar (interactive) method has been described by Dmitriev et al. [8], who utilized the periodicity properties of Halton sequences to identify the photon groups to update. Jiménez et al. [22] then generalized this approach to participating media, which however causes the algorithm to perform about an order of magnitude slower than necessary for interactive frame rates. Notable GPU implementations of photon mapping were proposed by Purcell et al. [36] and Zhou et al. [47], but only Krüger et al. [27] consider participating media. Photons can be splatted to surfaces efficiently using the GPU instead of performing costly density estimations [33]. We generalize this idea to volumetric photon mapping.

Interactive methods for general media. In recent years the increasing computational power of modern GPUs enabled development of interactive approaches for participating media rendering, although they are still burdened by various limitations.

Approaches that consider only a subset of light paths in the medium are also common, partly because they usually map well to modern GPU architectures. Often they restrict themselves to modelling single scattering, even when including advanced effects such as volumetric caustics [45, 10, 2, 29]. Ihrke et al. [17] use wavefront rendering to evaluate the Eikonal equation in volumes, considering refraction and single scattering in heterogeneous media. Kaplanyan and Dachsbacher [23] use a discrete-element approach similar to lattice-Boltzmann lighting in order to render low-frequency global illumination and low-order scattering. Shadows for single-scattering can efficiently be computed using deep shadow maps [30], which generalize binary shadow maps to storing a transmission function per pixel. Yet again, these methods are unsuitable for clouds rendering — that is simply because they neglect the higher scattering orders, which play a significant role for the appearance of clouds.

Engelhardt et al. [11] present a method based on the original instant radiosity approach by Keller [24]. Although successfully capturing appearance of several participating media, it shares the limitations of the original instant radiosity approach, namely the difficulty with handling highly anisotropic reflections and scattering.

Precomputation and Caching. Another possible approach is caching of illumination globally for the entire volume [18, 34]; various methods from this group then differ primarily in how this is achieved. Kautz et al. [39] include simple participating media in their PRT approach, while Zhou et al. [48] use a decomposition of the medium into a low frequency representation and a high-frequency residual field, and solve the radiative transport in the medium by applying the diffusion equation to the low-frequency field. However, these approaches rely significantly on precomputations and therefore are not suitable for dynamic media. Moreover, the illumination information is encoded into spherical harmonic functions, which are not well suited for representing highly anisotropic angular distributions.

Cloud rendering. Although specialized methods for rendering of clouds are more typical in the interactive domain, early non-interactive methods focused on simulating this phenomenon exist. Examples of these include works by Gardner [13] and Nishita et al. [35]. These clearly demonstrate that clouds have always been of prime interest among participating media.

Though many diverse approaches for interactive clouds rendering exist, they can roughly be categorized as empirical and physically-based. From the former group we mention the typical billboard-based technique by Wang [42] used for instance in Microsoft Flight Simulator 2004 or CryEngine2 [43].

On the other hand, many different paradigms have been taken in the existing physically-based approaches. Riley et al. [38] use the half-angle slicing technique of Kniss et al. [25] known from the domain of volume visualization. The method, however, is still semi-empirical and considers only the forward portion of multiple scattering and a single light source. The concept of Monte-Carlo illumination networks by Szirmay-Kalos et al. [41] is more theoretically sound, its main drawback however is the necessity to recalculate the entire network from scratch every time the cloud density field changes. Yet another approach has been taken by Bouthors et al. [3, 4] who analyse light behaviour in plane-parallel homogeneous slabs and based on this analysis design a series of *ad-hoc* functions to obtain illumination in the rendered cloud. The method produces interesting results, it is however limited to simple light sources and builds primarily on theoretical assumptions rather than on actual observations of the simulated environment.

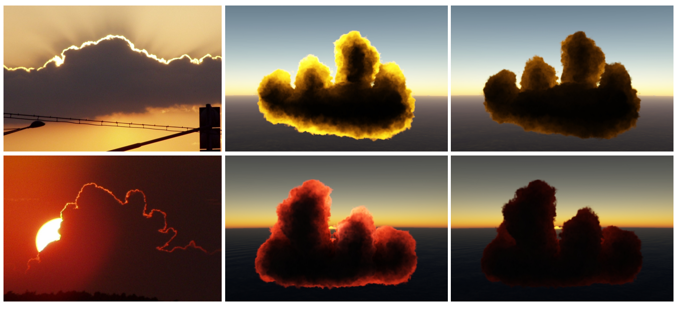

Figure 3: Anisotropic scattering is a defining visual property of clouds, causing above all the *silver lining* effect. Without correctly handling anisotropic scattering it is difficult to reproduce this phenomenon.

Discussion. Generally, we observe that the existing methods for rendering of participating media can be categorized in yet another way. The first category are approaches that can very rapidly simulate single scattering, but handle multiple scattering either in a supplemental way or even not at all. On the other hand, approaches that handle multiple scattering using lattice-based propagation or diffusion techniques fail at reproducing low-order or highly anisotropic effects (Fig. 3). The main difficulty of clouds lies in the fact that their high scattering anisotropy makes the transition from low-order to high-order scattering effects very gradual (see Fig. 2), thus these effects need to be handled in a consistent manner. The third group of hybrid methods that use ray-based propagation of radiant energy, but cache this energy in some way, has the potential to handle the described phenomenon. Unfortunately, this is usually hindered by the use of spherical harmonics to store the angular radiance distributions, destroying any high-frequency illumination effects (or making the caching process too costly as the number of basis coefficients needed to represent increasing frequencies grows quadratically). To remedy this situation is the main target of our work.

3. Method overview

This section provides an overview of our method. We first introduce the physical model and assumptions we use (Sec. 3.1) and then give a brief overview of the method itself (Sec. 3.2).

3.1. Model

The physical model that describes the propagation of light through participating environments is mathematically described by the radiative transfer equation (RTE) [7, 31]:

$$\frac{dL(\mathbf{x},\omega)}{d\mathbf{x}} = -\sigma_t(\mathbf{x})L(\mathbf{x},\omega) + \sigma_s(\mathbf{x})\int_{\Omega_{4\pi}} p(\mathbf{x},\omega,\omega')L(\mathbf{x},\omega')\,d\omega'.$$

RTE describes differential change of radiance L at a point in space $\mathbf{x}$ in the direction ω. σ_t and σ_s are the extinction and scattering cross sections of the simulated medium (in clouds we consider $\sigma_t = \sigma_s$). The spherical function p is the phase function; if p is the same for the entire medium the first parameter $\mathbf{x}$ can be omitted. In addition, p is usually considered to be rotationally symmetric around the incident direction ω and therefore becomes a 1D function of the angle between the incident and the outgoing direction. We employ both of these assumptions and will therefore use the notation $p(\theta)$ interchangeably with $p(\omega,\omega')$. Since we do not consider emission in clouds our formulation of RTE does not contain an emission term.

The classical volumetric photon mapping solves the RTE by performing a random Monte Carlo walk through the medium. This however requires the ability to determine the next location where the simulated photon will interact with the medium. To obtain this location in an unbiased manner we utilize the Woodcock tracking technique [44]. Please refer to Appendix A for its brief description.

The phase function model we employ is the standard formulation by Henyey and Greenstein [16]:

$$p_{HG}(\theta|g) = \frac{1}{4\pi}\cdot\frac{1-g^2}{(1+g^2-2g\cos\theta)^{3/2}} \qquad (1)$$

where $g \in (-1,1)$ denotes the anisotropy of p_{HG}. It is an average weighted cosine of the scattered directions produced by any given p:

$$g = \int_{\Omega_{4\pi}} p(\theta)\cos\theta\,d\omega'. \qquad (2)$$

There are multiple reasons why we have decided to use the Henyey-Greenstein function instead of a tabulated rigorous Mie phase function (such as for instance in Bouthors et al. [3, 4]). The Henyey-Greenstein function thanks to its closed analytical form is more portable, cheaper to evaluate and can be efficiently importance-sampled. Moreover, we utilize its mathematical properties to represent the illumination in clouds as well (Secs. 4.1.2 and 4.4). The main drawback of using this approximation is its inability to reproduce advanced scattering effects occurring in clouds, e.g., fogbows and glories [4]. However, these phenomena are comparatively rare and can be modelled separately.

Assumptions. At this point we establish assumptions about the environment we aim to simulate. We base these assumptions primarily on actual observations of the simulated environment, and later demonstrate how they are used to the method's advantage.

- **Illumination.** Clouds are virtually always illuminated by a single strong yet slowly-moving light source with one *dominant direction* (the Sun or the Moon) and by additional slowly-changing environmental light sources (the sky, street lights in urbanized areas, etc.). Occurrences such as an airplane rapidly flying through a cloud are sufficiently rare and we deem acceptable to neglect them. Mathematically, this assumption can be expressed in the way that clouds are illuminated by low-frequency light sources, and that both in temporal and spatial domain.
- **Mass distribution.** Clouds are indeed dynamic media, as their shape changes due to air convection. However, not only is this process never very rapid, it is seldom even noticeable for a human observer. Movement of whole clouds across the sky due to wind is usually relatively slow as well.

Apart from these, no other assumptions are made by our approach. We allow for any type and number of light sources, as long as there is one with a dominant radiant power and direction. Even local light sources are acceptable, if they move slowly. Similarly, we allow for arbitrary cloud shapes that can be represented by a 3D discrete density field. The source of this density field can be arbitrary as well — it can be stored as a series of volumetric animation frames in memory, or can result from a dynamic cloud convection simulation. No precomputations need to be performed on these data. And finally, we do not impose any limitations on the observer orientation or movement.

3.2. Overview

The input to our approach is the scalar density field of the simulated cloud, stored in a regular grid $\mathbf{D}(i,j,k) : \mathbb{N}^3 \to \mathbb{R}^+$ (see Fig. 4). Assuming a slowly changing environment we cache illumination into another regular grid $\mathbf{I}(i,j,k) : \mathbb{N}^3 \to \mathbb{R}^4$. Every cell in $\mathbf{I}$ stores the RGB flux as well as one anisotropy coefficient (Eq. 2) to model the light distribution (Sec. 4.1.2). The grid $\mathbf{I}$ is updated progressively, so each cell stores a mix of the current and several increasingly outdated values; however, as long as our assumptions hold, the difference from the ideal solution will be very small.

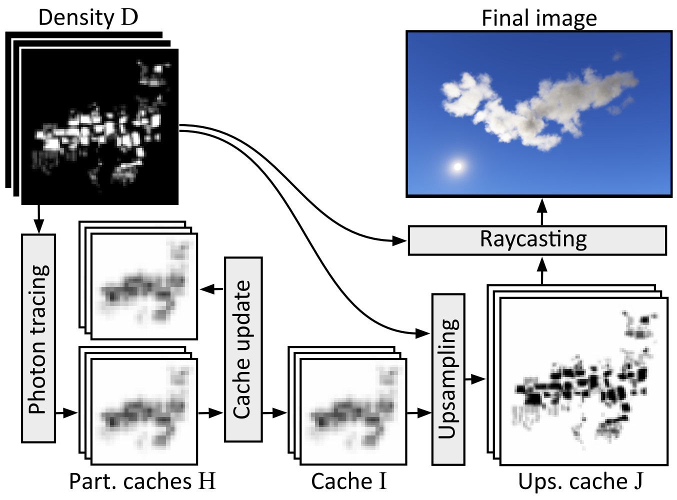

Figure 4: Schematic view of our algorithm pipeline.

The caching data structure is a circular buffer of m (we use $m = 100$) partial caches $\mathbf{H}_0(i,j,k),\ldots,\mathbf{H}_{m-1}(i,j,k)$ that combine as

$$\mathbf{I}(i,j,k) = \frac{1}{m}\sum_{l=0}^{m-1}\mathbf{H}_l(i,j,k)$$

to the full cache. The spatial resolution of the cache $\mathbf{I}$ is much lower than the one of the density field $\mathbf{D}$ to keep the amount of consumed memory and the variance of $\mathbf{I}$ in low values. To improve quality, before reconstructing the final image from $\mathbf{I}$ and $\mathbf{D}$ using ray-marching we upsample $\mathbf{I}$ to a second high-resolution cache $\mathbf{J}$, which has the same resolution as $\mathbf{D}$.

The global energy state of the cloud is represented by n_t photons, which are divided into m generations, each containing $n_\mathrm{g} = n_\mathrm{t}/m$ photons. In each frame f the algorithm performs the following:

1. The flux of n_g photons that are shot and traced through $\mathbf{D}$ is stored into a *new* partial cache $\mathbf{H}_\mathrm{new}$ (Sec. 4.1).
2. The partial cache $\mathbf{H}_\mathrm{old} = \mathbf{H}_{f \bmod m}$ in the circular buffer becomes the *old* partial cache and we replace it with the new partial cache $\mathbf{H}_\mathrm{new}$ as: $\mathbf{I}^{(f)} = \mathbf{I}^{(f-1)} - \mathbf{H}_\mathrm{old} + \mathbf{H}_\mathrm{new}$ (Sec. 4.2).
3. The low-resolution cache $\mathbf{I}$ is upsampled to another cache $\mathbf{J}$ using the density field $\mathbf{D}$ as a guidance signal (Sec. 4.3).
4. Ray-marching $\mathbf{D}$ and $\mathbf{J}$ produces the final image (Sec. 4.4).

In the following we present the details of each step.

4. Our Approach

In this section we describe temporally-coherent volumetric photon mapping in more detail. We will describe it for a general fine-grained parallel machine (our implementation uses CUDA). Such a machine executes many parallel threads, can read and write into buffers, read from special buffers called "textures" that provide efficient one-, two- and three-dimensional linear filtering, and can read and write to fast "local" memory.

At the beginning of each frame the cloud density field $\mathbf{D}$ is updated and stored into a texture, e. g., using a simulation of cloud dynamics or by streaming the dataset from memory on-the-fly.

4.1. Illumination

Illumination is computed by tracing photons (Sec. 4.1.1) in the density field $\mathbf{D}$ and storing (Sec. 4.1.2) them into a new partial cache $\mathbf{H}_{\mathrm{new}}$. The spatial resolution of $\mathbf{H}$ is typically much lower than of $\mathbf{D}$: the number of its cells should not exceed 2400 (for a cubic grid this limits the resolution to 13^3) in order to fit into the 48 kB of shared local memory available on current GPUs.

4.1.1. Photon tracing

In every step, n_g photons are shot from the Sun and the sky (note that different number of photons may be traced in each frame, e. g., if there is a need to balance time taken by other tasks performed by the GPU; the only requirement is that these photons carry the same flux, so that the energy state of the scene remains consistent). First, they are attenuated due to atmospheric scattering using a tabulated model [5, 9]. For photon propagation, we adapt the *photon marching* technique presented by Jarosz et al. [19]: for every photon a parallel thread is started and runs in a while-loop that scatters and stores photons (Sec. 4.1.2) until the photon leaves the volume. Instead of depositing photons in constant [19] or adaptively-sized [28] steps, we perform randomly-sized steps – since we employ Woodcock tracking (that itself performs randomly-spaced steps to generate interactions with the medium) for obtaining free photon path through the medium, we deposit photons at the locations where Woodcock tracking examines the currently generated event. This essentially helps us to avoid two simultaneous stepping procedures along the propagation ray. See Fig. 5 for an illustration of this process.

The downside of such an approach is of course the potentially higher variance of the solution. However, as we aggregate the photon contributions into the grid cells (which are much larger than the average mean free path in the medium) this effect will likely be very small.

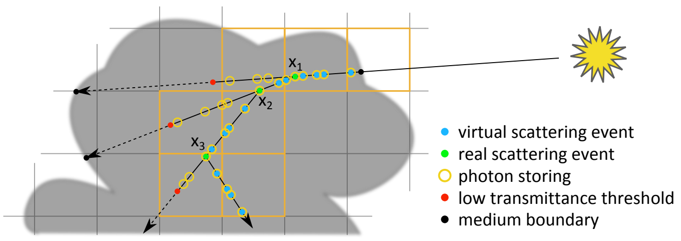

Figure 5: The scheme of our photon beam propagation procedure. Only the orange-framed grid cells receive photon energy. For details please refer to Appendix A.

In addition, the original photon marching technique traces each photon beam up to the medium boundary. Such an approach is indeed suitable for optically thin media such as fog and certain kinds of smoke which Jarosz et al. [19] use in their demonstrations. However, in optically thick environments, such as clouds, this approach invests large amounts of computational effort into simulating and storing photons that—due to low medium transmittance—carry only very little energy. To overcome this problem we simply stop tracing a beam if its transmittance gets below a small threshold. Although this of course introduces a slight (controllable) bias, our experiments show that the resulting speed-up can even reach and order of magnitude.

Further, we use the similarity theory [46] to speed up the tracing process. We pose a fixed threshold $t = 0.05$ and if the cumulated scattering anisotropy of a traced photon gets below t, we switch to the reduced scattering cross section[1] $\sigma_s' = \sigma_s\sqrt{(1-g)}$ and isotropic phase function. In our case it is easy to determine this, because the angular distribution of light scattered i times corresponds to i self-convolutions of the phase function, which for Henyey-Greenstein function in turn corresponds to just using g^i instead of g as its anisotropy parameter [32]. Therefore for a given photon we switch to σ_s' and isotropic scattering after $i = \log(t)/\log(|g|)$ bounces.

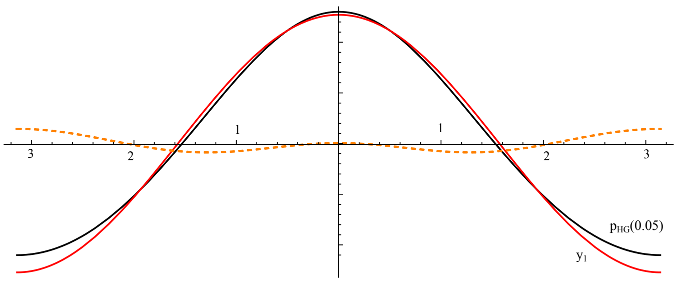

Figure 6: Comparison between p_{HG} with $g = 0.05$ and the first order spherical harmonic function y_1. The dashed orange plot represents the difference between these two functions.

The choice of t is not arbitrary; according to the derivation of the similarity theory [46] the similarity relations are valid if the radiance anisotropy is linear, i. e., if it can be expressed by spherical harmonic functions with the maximum degree of one. Except for the case of $g = 0$ this is unfortunately not possible for p_{HG}. We can therefore aim at least for a difference within some error threshold (see Fig. 6). The error of 2 % traditionally used in computer graphics corresponds roughly to $g = 0.1$; however, we need to reach a lower error margin as the error accumulates with each scattering bounce. Using $t = 0.05$ ensures an approximately five times lower error margin, which we deem sufficient for our application.

4.1.2. Photon storing

The classical (volumetric) photon mapping [21] and volume irradiance caching [18] both have two shortcomings that hinder their application to interactive cloud rendering: maintaining and querying of complex spatial data structures and limited support for highly-anisotropic scattering due to spherical harmonic functions used for the caching. Our approach overcomes both limitations by using a simple regular grid and a different basis function.

[1]We would like to point to the fact that this formula is different from the one used in many other computer graphics papers; the traditionally used one does not contain the square root and is incorrect according to the original derivation of Wyman et al. [46].

Regular photon grids. Classic volumetric photon mapping stores photons into hierarchical data-structures, such as kD-trees, that can adaptively resolve fine spatial details in the photon distribution as found, e. g., in occlusions or caustics in surface lighting. For clouds however, fine spatial details are usually not that prominent. On the other hand, inserting photons into a complex structure and performing an adaptive density estimation is less suitable for contemporary massively-parallel machines which we target. We therefore suggest to trade adaptivity for simplicity and revert to a plain regular grid in which the illumination is accumulated, similar to irradiance volumes [15] (also see Fig. 5). This grid **H** is sufficiently small to fit into local memory shared by a block of threads executed on one multi-processor. Storing a photon into such a three-dimensional grid is now as efficient as splatting it in two dimensions [33]. Also, as atomic operations are necessary to perform the accumulation properly, for the fast local shared memory we found them not to cause virtually any additional overhead.

Sec. 4.3 explains how an upsampling scheme can be used to improve the effective spatial resolution of the cache.

Henyey-Greenstein basis. Approaches that cache the illumination information inside the volume [18, 34] use spherical harmonics to approximate the radiance function. This is suitable for isotropic and moderately anisotropic media, but not for strongly forward-scattering media such as clouds, where a very high number of SH coefficients is required. This is problematic, as the strong forward scattering in clouds is visually important for their appearance, mainly causing the well known *silver lining* phenomenon (Fig. 3).

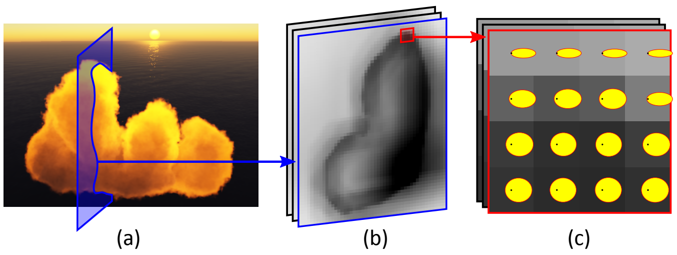

Figure 7: Local Henyey-Greenstein lobes – (b) shows one slice of the greyscale-encoded anisotropy coefficients of the illumination solution that correspond to the marked region on the cloud (a). The exact shape of the lobes in the cache locations is shown in (c). The light in (c) is coming from the left.

To overcome this limitation we propose to represent caches using the Henyey-Greenstein function (Eq. 1). Evidence shows [17] that in situations where most energy in scene comes from a single direction then also a majority of scattered light will intuitively propagate in a very similar direction, which will be especially true for clouds and other media with high scattering anisotropy. Thus, when storing a photon into a cache cell, in addition to its radiant flux we also accumulate its cosine with respect to the dominant light direction. This only requires to store four values (an RGB-triple for flux and a single cosine value) per cell and the projection consists of evaluating just a single dot product per photon. In addition we count the number of photons arriving at each cell, and after all photons are shot, divide the accumulated cosines by the number of contributing photons for a proper normalization. The value obtained this way is anisotropy factor for the given cache cell (Eq. 2). We can then directly use the Henyey-Greenstein function to represent the per-cell angular illumination distribution (Fig. 7).

Photons which are not emitted by the Sun but come from the sky are not projected into the HG basis. As they—in difference from the Sun photons—arrive from a wide distribution of directions, their angular contribution to the caches will arguably be close to uniform. In case this does not hold (for instance if the sky is much brighter than the ground reflection) it is easy to add an additional HG lobe and obtain the reference direction by weighting the sphere of directions by the relative energy contributions of the environmental lighting.

4.2. Update

After computing the cloud illumination using the photon tracing explained before, the partial cache $\mathbf{H}_{\text{new}}$ is added to the global solution **I** and the outdated cache $\mathbf{H}_{\text{old}}$ is removed. This is done using a simple parallel addition and subtraction over all cache cells.

4.3. Upsampling

At this point the cache **I** could already be used for rendering. However, its spatial resolution is usually much lower than the one of **D**. This was required to fit it into shared local memory and to reduce variance of the illumination solution (see Appendix B), as well as the overall memory footprint. This, however, has two problems. First, the low resolution of **I** can cause a substantial blur of the light energy and cause interpolation artefacts. Second, as the relation between the anisotropy factor g and the resulting lobe shape is quite nonlinear, linearly interpolating it during the cache fetching would sometimes produce incorrect angular distributions of the scattered radiance.

To avoid these limitations, we propose to upsample **I** to **J**, a cache with the same spatial resolution as **D** (Fig. 8). One possible approach to this is to apply the joint bilateral upsampling [26]. The idea of the joint bilateral upsampling is to use a bilateral filter consisting of two parts: a traditional domain filter, e. g., Gaussian, and a range filter operating above a *guidance signal*, which provides an additional regulatory mechanism over the filtering process. In our case it is very natural to use **D** as the guidance signal. This approach produces empirically good results, which is not surprising as it can be assumed that in clouds the density field **D** and the scattered radiance function L_{out} (Sec. 4.4) will spatially correlate on a local scale (roughly within one photon mean free path distance).

The main issue with the joint bilateral upsampling is that it employs Gaussian filters, as mentioned above. This requires specification of their parameters – standard deviations and, in case of their truncated versions, also effective radii. It is not necessarily clear what these values should be, and they can moreover vary not only for different clouds, but also for a single cloud during its evolution. Since they also do not directly relate to the physical properties of clouds, these can not be used to derive them in a natural way either.

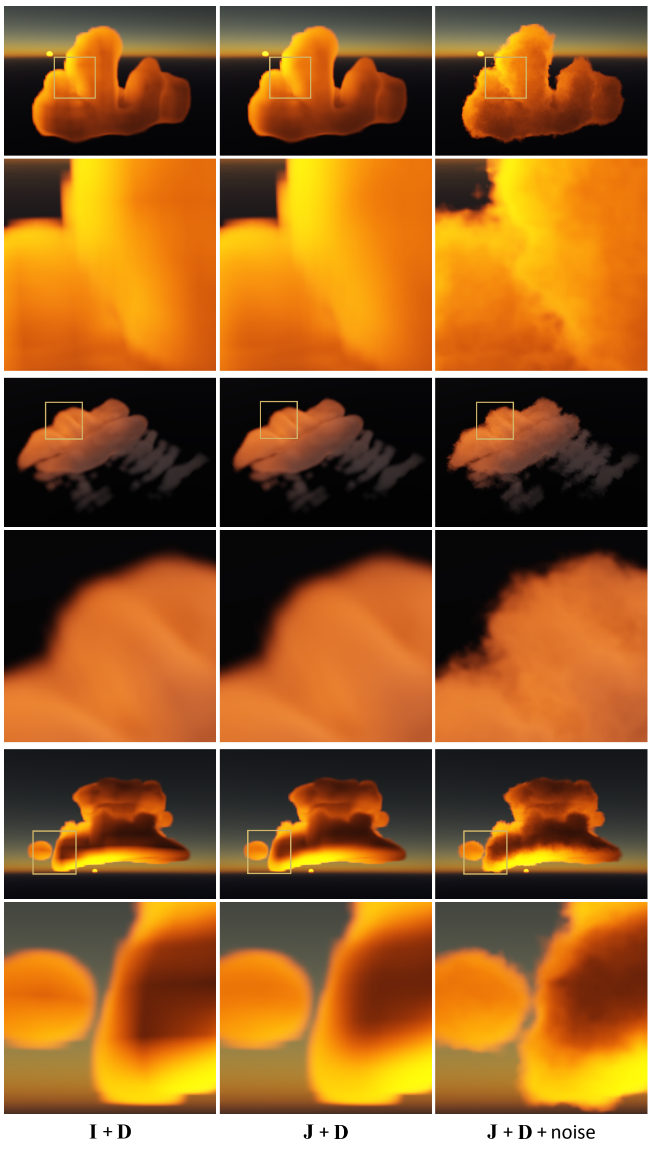

Figure 8: Upsampling of the coarse illumination solution **I** to **J** removes a majority of artefacts caused by linear interpolation and incorrect handling of the radiance anisotropy. The additional noise perturbation helps to break the unnatural smoothness of the underlying density field **D**.

To overcome this problem it is useful to exploit the additional knowledge we have available here, both in terms of the working data and the fact that the upsampled quantities have direct physical meanings. Based on this knowledge we can then design a more suitable physically-plausible upsampling filter.

At the end of the tracing process **I** contains two different quantities that need to be upsampled: radiant energy density and its angular distribution represented by the HG anisotropy factor.

Radiant energy. From the RTE we can see that the radiant energy L is locally proportional to the medium scattering cross section, which in turn is linearly dependent on the density field **D**. Our proposition for its upsampling is using the joint bilateral filter f_L as follows. The domain filter needs to take into account that L is exponentially attenuated in space due to the medium transmittance. The range filter, which naturally uses **D** as the guidance signal, has to incorporate the linear dependence of the scattered L on the medium density. Mathematically this can be written as follows:

$$f_L(d_\mathrm{D}, d_\mathrm{R}|\tau, r) = f_\mathrm{texp}(d_\mathrm{D}|\tau, r) \cdot f_\mathrm{lin}(d_\mathrm{R})$$

$$f_\mathrm{texp}(d|\lambda, r) = \max\left(\frac{f'_\mathrm{texp}(d|\lambda, r) - f'_\mathrm{texp}(r|\lambda, r)}{1 - r \cdot f'_\mathrm{texp}(r|\lambda, r)}, 0\right)$$

$$f'_\mathrm{texp}(d|\lambda, r) = \frac{\lambda \cdot \exp(-\lambda d)}{1 - \exp(-\lambda r)}$$

$$f_\mathrm{lin}(d) = \max(1 - d, 0)$$

where d_D and d_R are the distances in the spatial and range domain respectively, τ is the optical thickness in the medium and r is the effective filtering radius (please see below for the definition of these parameters). The reason for this formulation is the infinite support of the exponential distribution (which should correctly be used). As we are using only a local neighbourhood for filtering, using the exponential distribution would introduce discontinuities in the upsampled solution **J**. We therefore use a truncated exponential distribution f_texp, which we define by normalizing the thresholded exponential distribution[2] f'_texp.

Radiance anisotropy. As already mentioned the upsampling of the anisotropy g' has to take into account the nonlinear relation between its value and the distribution it represents. However, there is no straightforward way to compensate for this nonlinearity without an additional information about the medium.

Sec. 4.1.1 describes how the radiance anisotropy changes by light scattering – when the light progresses into the medium, its anisotropy decreases with the increasing number of scattering orders. The anisotropy at some point will therefore well correlate with the *average scattering order* at that point. This quantity, which we call *penetration depth* γ, can easily be obtained during the tracing process; we track the number of times a traced photon scattered, and store this value along with the photon's energy and average cosine. γ already is close to linear, and thus can be linearly interpolated from the low-resolution grid during the upsampling. The upsampling filter for g' can then be defined as

$$f_{g'}(\delta|g) = g^\delta$$

where δ is the difference of penetration depth γ between the upsampled point and the point from which we are extrapolating the anisotropy value, and g is the medium scattering anisotropy. The logic behind this filter is exactly the same as in the similarity heuristic in Sec. 4.1.1. Note that δ can also have a negative value, which means that if the upsampled point has a lower penetration depth than the extrapolated point, its anisotropy will be higher.

Final upsampling. The upsampling procedure itself then iterates through a local neighbourhood Δ of the upsampled point, extrapolating the contributions from the neighbouring points to

[2] f'_texp parametrized by r is an exponential distribution modified to have a unit integral on the interval $\langle 0, r\rangle$.

its radiant energy and anisotropy:

$$\mathbf{J}_L(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Delta} \mathbf{I}_L(\mathbf{q}) \cdot w_L}{\sum_{\mathbf{q}\in\Delta} w_L} \qquad \mathbf{J}_{g'}(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Delta} \mathbf{I}_{g'}(\mathbf{q}) \cdot w_{g'} \cdot w_L}{\sum_{\mathbf{q}\in\Delta} w_{g'} \cdot w_L}$$

$$w_L = f_L(d_\mathrm{D}, d_\mathrm{R}|\tau, r) \qquad w_{g'} = f_{g'}(\delta|g)$$

and $d_\mathrm{D} = \|\mathbf{p}-\mathbf{q}\|$, $d_\mathrm{R} = |\mathbf{D}(p) - \mathbf{D}(q)|$, $\delta = \mathbf{I}_\gamma(\mathbf{p}) - \mathbf{I}_\gamma(\mathbf{q})$, $\tau = \sigma_\mathrm{s} \cdot (\mathbf{D}(p) - \mathbf{D}(q))/2$ and r is the shortest distance from the centre of Δ to its boundary.

It can be seen that we multiply the weights $w_{g'}$ for upsampling the anisotropy by w_L. This is because the points in the cache which contribute more energy to the upsampled point should also have a higher overall impact on its resulting anisotropy.

The upsampling maps well to GPUs – it is a purely local operation executed in parallel exactly once per each upsampled solution cell. According to our experiments using a 3^3 neighbourhood already produces good results. The upsampling could also be performed directly during the ray-marching, but we found it to be much faster when done in an intermediate step, as this allows more coherent reading of **I** and **D** and no work is duplicated.

4.4. Ray-marching

To visualize clouds we use the standard ray-marching [28] with early-exit at 2 % transmittance threshold. For every pixel, a thread is started that marches **D** and **J**, accumulates radiance from **J** (see below) and transmittance T from **D** in front-to-back order, applies compositing accordingly and terminates when T is less than 2 %.

To keep the memory consumption in manageable levels we use only moderate resolutions for **D** (see Table 1). We then increase the amount of detail by procedural perturbation of the texture coordinates for fetching **D** and **J** by three octaves of simple vector noise stored in a small 3D texture, similar to Kniss et al. [25] (Fig. 8).

For most clouds it is not necessary to ray-march them in full screen resolution, as the frequency of the applied noise is still lower than the screen sampling frequency (see Fig. 9, left). Therefore in all cases we render the cloud in half the screen resolution, in both dimensions.

In addition, because of the good temporal coherency of the environment, it is wasteful to conservatively ray-march the cloud volume in every frame. We have therefore implemented an impostor caching to avoid this. Our proposition is to use several empirical criteria, which, if any is exceeded, indicate that the cloud impostor should be updated:

1. Angle criterion – if the relative angle under which the cloud is seen changes by more than 5 degrees.
2. Distance criterion – if the relative distance from the cloud changes by more than 20 %.
3. Illumination criterion – if the photon map has been updated by more than 4 % (with respect its update period m).
4. Animation criterion – if the cloud shape changed significantly (this is of course dependent on the cloud evolution speed and needs to be tuned individually).

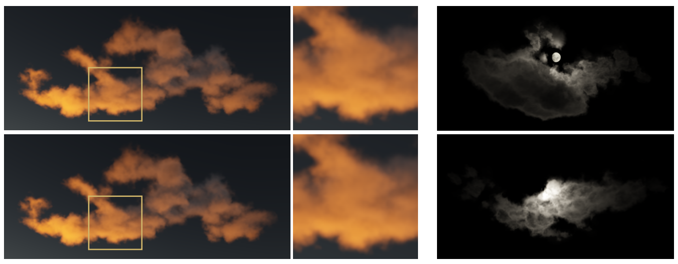
Figure 9: Left: Cirrocumulus altocumulus rendered at full resolution (top, ray-marching time 77 ms) and half resolution (bottom, ray-marching time 21 ms). Right: clouds rendered at night.

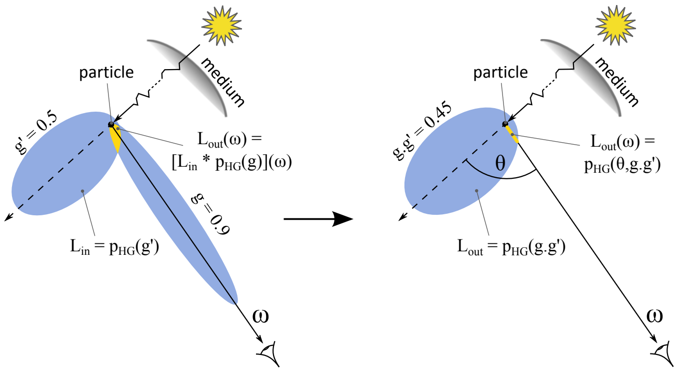

Figure 10: Two possible ways of reconstructing the illumination information represented by the Henyey-Greenstein function. Left: the yellow area represents the value of $L_\mathrm{out}(\omega)$ obtained by convolving L_in and $p_\mathrm{HG}(g)$. Right: the yellow segment is directly the value of $L_\mathrm{out}(\omega)$.

As will be seen in Sec. 5, the two above optimizations increase the ray-marching speed by about an order of magnitude, and that without causing visual degradation.

Reconstruction. The illumination reconstruction is straightforward at this moment. **J** contains in each cell RGB intensities of the directional and environmental illumination energy and the local directional anisotropy factor g', which represents the angular distribution of this energy.

It is however important to note that since we use the incident photon direction for the projection step during the photon tracing procedure (Sec. 4.1.1), the value of g' represents the *incident* radiance function L_in at the cache location. To obtain the desired *outgoing* radiance value $L_\mathrm{out}(\omega)$ for a given view direction ω it is necessary to convolve L_in (represented by $p_\mathrm{HG}(g')$) with the medium phase function centred around the viewing direction (Fig. 10, left). Instead of computing the convolution directly (e. g., numerically) we once more utilize the self-convolution property of the Henyey-Greenstein function (Sec. 4.1.1) – and simply evaluate $p_\mathrm{HG}(\theta, g \cdot g')$, where g is the global anisotropy factor used for scattering simulation in the cloud (Fig. 10, right). This is mathematically equivalent (see Appendix C).

An additional benefit of our Henyey-Greenstein representation is that the angular illumination information represented this way reproduces the scattering anisotropy in clouds well, but at the same time is smooth. This prevents emergence of high-frequency noise, which would look unnatural in clouds.

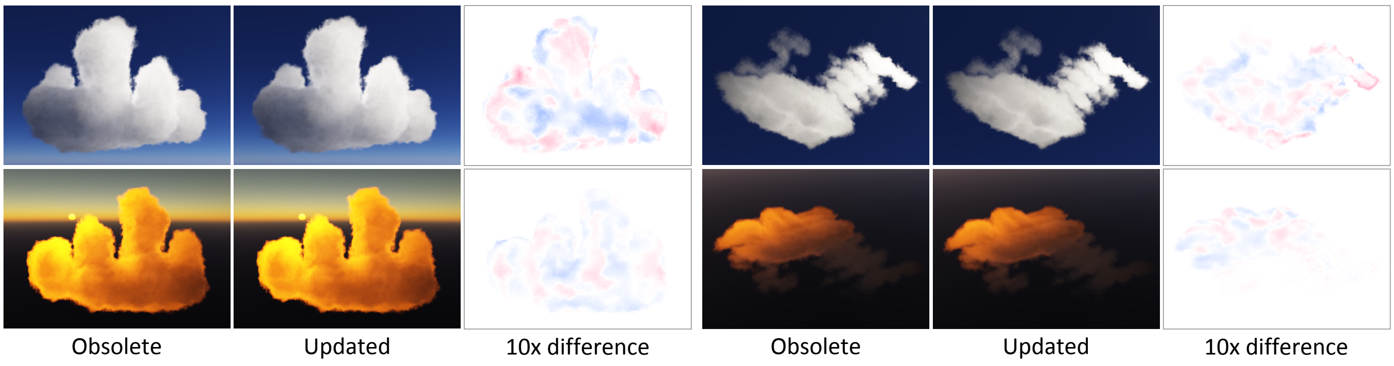

Figure 11: Illumination obsolescence in Cumulus congestus and Altostratus undulatus in two different times of day. The obsolete cases were captured during a continuous Sun movement of 0.5 deg / s.

5. Results and Discussion

Our tests have been conducted on a laptop PC with 2 GHz Intel Core i7-2630QM CPU, NVidia GTX 485 Mobile GPU and Windows 7 64-bit. In all our tests we use the following global settings: 819 k photon beams, $m = 100$ partial caches, ray-marching step size of $d/300$ (where d is the cloud bounding box diagonal) and a screen resolution of 1920×1200. All density fields beside the PBRT smoke were modelled by hand as meshes and voxelized into 3D grids. We have performed three kinds of tests – method performance measurements, analysis of the illumination outdating during scene state change, and rendering quality in dependence on the photon map resolution.

Performance. We have used three datasets for timing measurements: static datasets of Cumulus congestus (dense concentrated cloud) and Altostratus undulatus (sparse wavy cloud) and the animated smoke dataset available in PBRT renderer (`pbrt.org`). The scene settings and measured timings are summarized in Table 1. Figs. 1, 9 (right) and 13 show the results of our method, including the measured datasets.

As Table 1 shows, the photon tracing costs are in the order of milliseconds, since only a low number of photon beams is traced per frame (about 8.2 k in our tests). Under such conditions, the ray-marching becomes the most expensive step, if performed conservatively in each frame. The impostor caching decreases the ray-marching costs by the factor of three, on average, which then also decreases the total rendering time significantly.

Illumination obsolescence. Since our method amortizes the illumination computation there is a various level of obsolescence in the partial solutions. To show that our update process is sufficiently fast we compare clouds rendered under constant light source movement against clouds under static illumination. Fig. 11 shows the resulting images. The maximal luminance difference was about 1.5 % throughout all cases, mostly due to a residual low-frequency photon noise. Please note that although the Sun's angular speed was only 0.5 deg / s this is still 120 times faster than the real speed and several times faster than is commonly used in interactive 3D applications (for instance the common time ratio in FPS games is 1 game-time hour to 2 real-time minutes, which corresponds to an angular speed of 0.125 deg / s).

Photon map resolution. Fig. 12 shows the Cumulus congestus dataset rendered with three different resolutions of the photon map: the resolution used in all other measurements, and two others, one with 8-times less and one with 8-times more cells. The photon tracing times naturally differ as well, because (as Appendix B describes) different amounts of photon beams have to be used to obtain coherent results. It can however be seen that for lower numbers of photon beams the simulation time does not scale down linearly. This is because if too few beams are traced simultaneously, the GPU utilization becomes too low. We therefore conclude that although the quality degrades rather gracefully with decreasing photon map resolution (mainly thanks to the upsampling stage), from a certain point doing so does not pay off anymore.

6. Conclusion and Future Work

We have presented an interactive cloud rendering method. Our algorithm utilizes a temporally-coherent illumination caching process to amortize the simulation costs across multiple frames. Our novel representation of angular illumination distribution inside clouds enables us to reproduce the characteristic appearance of many clouds, while keeping the computational and storage costs relatively modest. The algorithm maps very well to the architecture of modern graphics hardware, and hence all its major steps can be evaluated in parallel on the GPU.

The proposed method improves on the existing techniques in multiple regards. We build on realistic observations about the typical environment of clouds and use these to our advantage. Our method is not limited to any particular cloud type or any subset of possible light paths in clouds, and can handle the important light sources typical for the simulated environment. We consider all physically plausible properties of clouds and the entire method is physically meaningful to a large extent.

The possible future work includes several directions. First, there is need for a multi-resolution scheme to enable simulation of multiple clouds at once without linear decrease in speed. Second, the algorithm might possibly be adapted to other types of participating media by relaxing or even completely dropping the specific assumptions about the simulated environment. For example, multiple HG lobes could be used to represent more

Table 1: Performance of our algorithm in various stages. All other steps took less than 1ms to execute, thus we do not list them.

Dataset	**D** resolution	**D** size (km)	σ_s (m^{-1})	g	**I** resolution
Altostratus undulatus	70×22×71	1.4×0.4×1.4	0.03	0.96	15×10×15
Cumulus congestus	48×62×90	0.9×1.2×1.8	0.03	0.96	10×12×18
PBRT smoke (anim.)	100×40×100	2×0.8×2	0.03	0.96	15×10×15

Dataset	Photon tracing	Upsampling	Ray-marching (conservative)	Ray-marching (impostors)	Total (cons. / imp.)
Altostratus undulatus	6.1 ms	2.3 ms	22.2 ms	7.4 ms	30.6 ms / 15.8 ms
Cumulus congestus	9.1 ms	3.0 ms	20.7 ms	7.9 ms	32.8 ms / 20.0 ms
PBRT smoke (anim.)	7.8 ms	3.3 ms	33.2 ms	9.5 ms	44.3 ms / 20.6 ms

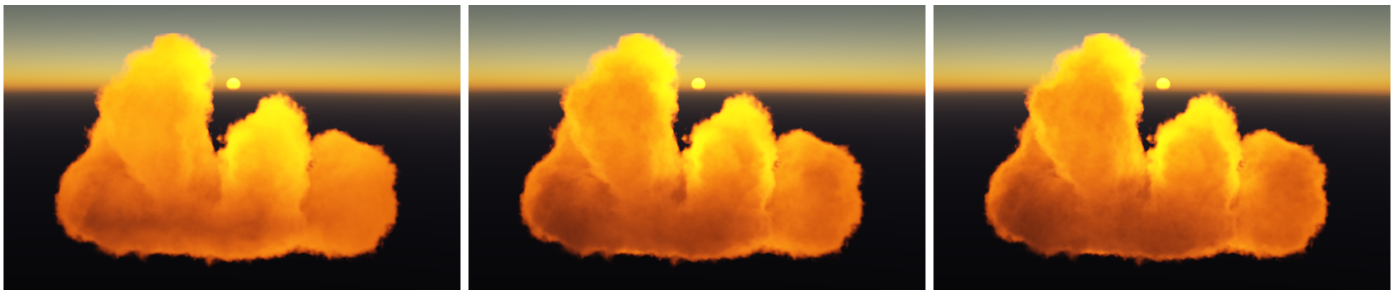

Figure 12: Quality comparison for different photon map resolutions, $m = 100$. Left: 270 cells (0.5 MB video memory), 102 k photon beams, 4.8 ms per frame. Middle: 2080 cells (4.2 MB video memory), 819 k photon beams, 9.1 ms per frame. Right: 16640 cells (33.8 MB video memory), 4096 k photon beams, 33.4 ms per frame. The middle image uses the settings used to generate all the other results for the particular dataset throughout the paper.

complex illumination distributions than the one dealt with in the described approach. It remains to be seen to what extent is this feasible.

Acknowledgements. We would like to thank Oliver Klehm, Jaroslav Křivánek and Karol Myszkowski for valuable discussions and the anonymous reviewers for their feedback. This work has been supported by the Max Planck Center for Visual Computing and Communication.

References

[1] Boris Airieau, Daniel Meneveaux, Flavien Bridault, and Philippe Blasi. Photon streaming for interactive global illumination in dynamic scenes. *Visual Computer*, 27:229–240, 2011.

[2] I. Baran, J. Chen, J. Ragan-Kelley, F. Durand, and J. Lehtinen. A hierarchical volumetric shadow algorithm for single scattering. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, volume 29, 2010.

[3] Antoine Bouthors, Fabrice Neyret, and Sylvain Lefebvre. Real-time realistic illumination and shading of stratiform clouds. In *Proc. Workshop on Natural Phenomena*, 2006.

[4] Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, and Cyril Crassin. Interactive multiple anisotropic scattering in clouds. In *Proc. I3D*, 2008.

[5] Eric Bruneton and Fabrice Neyret. Precomputed atmospheric scattering. In *Proc. EGSR*, 2008.

[6] Eva Cerezo, Frederic Pérez-Cazorla, Xavier Pueyo, Francisco J. Seron, and François Sillion. A survey on participating media rendering techniques. *Visual Comput.*, 21:303–328, 2005.

[7] Subrahmanyan Chandrasekhar. *Radiative transfer*. Dover Publications, 1960.

[8] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Proc. EGRW*, pages 25–36, 2002.

[9] Oskar Elek and Petr Kmoch. Real-time spectral scattering in large-scale natural participating media. In *Proc. SCCG*, pages 77–84, 2010.

[10] T. Engelhardt and C. Dachsbacher. Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. In *Proc. I3D*, pages 119–125, 2010.

[11] Thomas Engelhardt, Jan Novák, and Carsten Dachsbacher. Instant multiple scattering for interactive rendering of heterogeneous participating media. Technical report, Karlsruhe Institut of Technology, December 2010.

[12] Raanan Fattal. Participating media illumination using light propagation maps. *ACM Trans. Graph.*, 28:7:1–7:11, 2009.

[13] Geoffrey Y. Gardner. Visual simulation of clouds. In *Proc. SIGGRAPH*, pages 297–304, 1985.

[14] Robert Geist, Karl Rasche, James Westall, and Robert Schalkoff. Lattice-Boltzmann lighting. In *Proc. EGSR*, pages 355–362, 2004.

[15] G. Greger, P. Shirley, P.M. Hubbard, and D.P. Greenberg. The irradiance volume. *Computer Graphics and Applications, IEEE*, 18(2), 1998.

[16] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the Galaxy. *Astrophysical Journal*, 93(1):70–83, 1941.

[17] Ivo Ihrke, Gernot Ziegler, Art Tevs, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Eikonal rendering: efficient light transport in refractive objects. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26, 2007.

[18] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance caching for participating media. *ACM Trans. Graph.*, 27:7:1–7:11, 2008.

[19] Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph.*, 30:5:1–5:19, 2011.

[20] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proc. Eurographics)*, 27(2):557–566, 2008.

[21] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In *Proc. SIGGRAPH*, pages 311–320, 1998.

[22] Juan-Roberto Jiménez, Karol Myszkowski, and Xavier Pueyo. Interactive global illumination in dynamic participating media using selective photon tracing. In *Proc. SCCG*, pages 211–218, 2005.

[23] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation

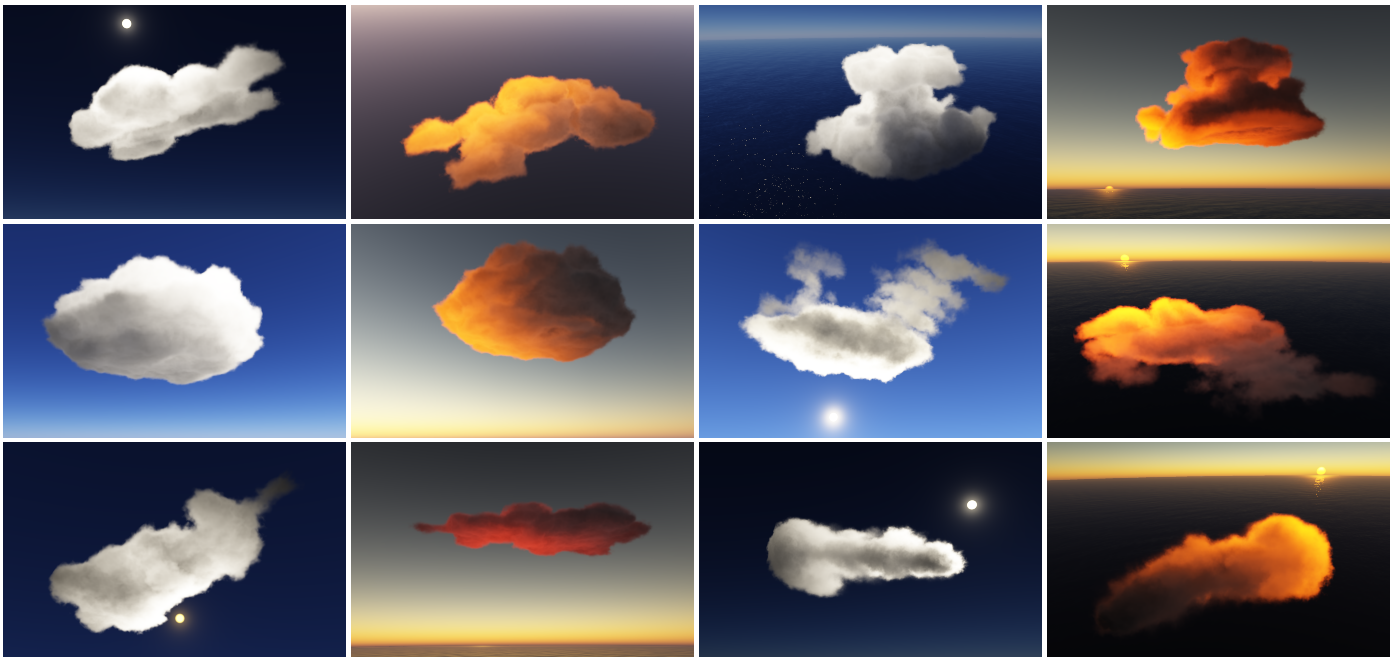

Figure 13: More results of our method. In reading order, by pairs: Cumulus humilis, Cumulonimbus incus, Altocumuls lenticularis, Altostraturs undulatus, Stratocumulus and the PBRT smoke dataset.

volumes for real-time indirect illumination. In *Proc. I3D*, 2010.

[24] Alexander Keller. Instant radiosity. In *Proc. SIGGRAPH*, pages 49–56, 1997.

[25] Joe Kniss, Simon Premoze, Charles Hansen, and David Ebert. Interactive translucent volume rendering and procedural modeling. In *Proc. Visualization*, pages 109–116, 2002.

[26] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3), 2007.

[27] J. Krüger, K. Bürger, and R. Westermann. Interactive screen-space accurate photon tracing on GPUs. In *Proc. EGSR*, 2006.

[28] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proc. IEEE Visualization*, 2003.

[29] Gabor Liktor and Carsten Dachsbacher. Real-time volumetric caustics with projected light beams. *Proc. I3D*, 2011.

[30] T. Lokovic and E. Veach. Deep shadow maps. In *Proc. SIGGRAPH*, pages 385–392, 2000.

[31] Nelson Max. Optical models for direct volume rendering. *IEEE Trans. Vis. and Computer Graphics*, 1:99–108, 1995.

[32] Nelson Max, Greg Schussman, Ryo Miyazaki, Kei Iwasaki, and Tomoyuki Nishita. Diffusion and multiple anisotropic scattering for global illumination in clouds. *J. WSCG*, 1–3, 2004.

[33] Morgan McGuire and David Luebke. Hardware-accelerated global illumination by image space photon mapping. In *Proc. HPG*, 2009.

[34] Jonathan T. Moon, Bruce Walter, and Steve Marschner. Efficient multiple scattering in hair using spherical harmonics. *ACM Trans. Graph.*, 27:31:1–31:7, 2008.

[35] Tomoyuki Nishita, Yoshinori Dobashi, and Eihachiro Nakamae. Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proc. SIGGRAPH*, pages 379–386, 1996.

[36] T.J. Purcell, C. Donner, M. Cammarano, H.W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proc. Graphics Hardware*, pages 41–50, 2003.

[37] Mathias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In *Proc. Monte Carlo and Quasi-Monte Carlo Methods*, pages 591–606, 2006.

[38] Kirk Riley, David S. Ebert, Martin Kraus, Jerry Tessendorf, and Charles D. Hansen. Efficient rendering of atmospheric phenomena. In *Proc. EGSR*, 2004.

[39] P.P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, volume 21, 2002.

[40] Joe Stam. Multiple scattering as a diffusion process. In *Proc. EGWR*, pages 41–50, 1995.

[41] László Szirmay-Kalos, Mateu Sbert, and Tamás Umenhoffer. Real-time multiple scattering in participating media with illumination networks. In *Proc. EGSR*, 2005.

[42] Niniane Wang. Realistic and fast cloud rendering. In *J. Graphics, GPU & Game Tools*, 2003.

[43] Carsten Wenzel. Real-time atmospheric effects in games. In *SIGGRAPH Course*, pages 113–128, 2006.

[44] E. Woodcock, T. Murphy, P. Hemmings, and T. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Application of Computing Methods to Reactor Problems*, pages 557–579, 1965.

[45] C. Wyman and S. Ramsey. Interactive volumetric shadows in participating media with single-scattering. In *Proc. IEEE Interactive Ray Tracing*, pages 87–92, 2008.

[46] Douglas R. Wyman, Michael S. Patterson, and Brian C. Wilson. Similarity relations for the interaction parameters in radiation transport. *Applied Optics*, 28(24):5243–5249, 1989.

[47] K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time kd-tree construction on graphics hardware. In *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, volume 27, page 126, 2008.

[48] Kun Zhou, Zhong Ren, Stephen Lin, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Real-time smoke rendering using compensated ray marching. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 36:1–36:12, 2008.

Appendix A. Woodcock tracking

One of the core operations performed by volumetric photon mapping is finding the distance to a next scattering or absorption event. We utilize Woodcock tracking to do this in an unbiased manner. Let us assume the last scattering event occurred at $\mathbf{x}_k$ and the generated (normalized) scattering direction according to p is ω. The next interaction will take place at $\mathbf{x}_{k+1} = \mathbf{x}_k + d_{\text{event}} \cdot \omega$. For heterogeneous media, to obtain the distance d_{event}

one needs to solve the implicit equation

$$\int_0^{d_{\text{event}}} \sigma_t(\mathbf{x}_k + t \cdot \omega)\, dt = -\ln(1-\xi)$$

where ξ is an uniformly distributed unit random variable.

Basically, Woodcock tracking steps along the photon propagation direction using steps with random exponentially-distributed length with the mean $1/\sigma_T$ (where σ_T is the peak extinction cross section value throughout the entire medium). The technique then in each step probabilistically decides if an interaction takes place at the given position (i. e., if the currently generated event is *real* or just *virtual*), so that the distance to the generated interaction event has the mean value of d_{event} as defined the above equation. Finally, Russian roulette with the probability of $\sigma_s(\mathbf{x}_{k+1})/\sigma_t(\mathbf{x}_{k+1})$ is used to decide if the generated interaction is scattering or absorption (this last step is skipped for clouds as $\sigma_t = \sigma_s$). For more details please refer to Raab et al. [37].

Appendix B. Temporal coherence

In order to avoid low frequency temporal noise in the rendered images it is necessary to maintain a low variance of the illumination solution $\mathbf{I}$. As the convergence rate of a Monte Carlo estimate is $O(1/\sqrt{n})$, to maintain 2 % error in each grid cell the total number of photons in scene n_t needs to be set so that each cell in $\mathbf{I}$ receives at least around $n \approx 2500$ photons (together from all partial caches $\mathbf{H}_0, \ldots, \mathbf{H}_{m-1}$). Since the photon budget is roughly fixed, the grid resolution needs to be adjusted accordingly to fulfill this criterion (in addition to the limitation imposed by the available shared memory of GPUs, see Sec. 4.1).

Note however that it is necessary to differentiate between photon beams and individual photons — a single photon beam can dispose many photons into the grid (Fig. 5). There is no precise way to determine the ratio between these two for a particular nontrivial dataset, but a good initial approximation is easily derived from the scattering cross section σ_s, the average density of the utilized dataset and its spatial dimensions.

Appendix C. Illumination reconstruction

Recalling the radiative transport equation described in Sec. 3.1:

$$\frac{dL(\mathbf{x},\omega)}{d\mathbf{x}} = -\sigma_t(\mathbf{x})L(\mathbf{x},\omega) + \sigma_s(\mathbf{x}) \int_{\Omega_{4\pi}} p(\mathbf{x},\omega,\omega')L(\mathbf{x},\omega')\, d\omega'$$

it can be seen from the definition that the integral term on the right is a spherical convolution between the incident radiance (hereinafter L_{in}) and the medium phase function p at $\mathbf{x}$.

In our approach we represent L_{in} by approximating the incident photons' flux by the p_{HG} basis function by averaging their cosine values in respect to the dominant Sun direction, thus producing an anisotropy factor g' representing the angular distribution of scattered photons at $\mathbf{x}$ (Sec. 4.1.2). Therefore $L_{\text{in}} \approx p_{\text{HG}}(g')$ and the integral term becomes

$$\int_{\Omega_{4\pi}} p(\omega,\omega')p_{\text{HG}}(\omega'|g')\, d\omega',$$

neglecting the location from now on. Note that the orientation of $p_{\text{HG}}(g')$ in space is given by the dominant light direction which is constant for the entire cloud (although nothing prevents the orientation from varying in space by simply storing the direction separately for each cell).

If the used medium phase function is also p_{HG} parametrized by some g, then utilizing the self convolution property of p_{HG} [32] (Sec. 4.1.1) we can write

$$\left[\int_{\Omega_{4\pi}} p_{\text{HG}}(\omega,\omega'|g)p_{\text{HG}}(\omega'|g')\, d\omega'\right](\omega) = p_{\text{HG}}(\omega|g \cdot g').$$

Thus $p_{\text{HG}}(g \cdot g')$ represents the outgoing radiance function:

$$L_{\text{out}} \approx p_{\text{HG}}(g \cdot g').$$