Eurographics Symposium on Rendering 2011
Ravi Ramamoorthi and Erik Reinhard
(Guest Editors)

Volume 30 (2011), Number 4

Progressive Expectation–Maximization for Hierarchical Volumetric Photon Mapping

Wenzel Jakob[1,2] Christian Regg[1,3] Wojciech Jarosz[1]

[1]Disney Research Zürich [2]Cornell University [3]ETH Zürich

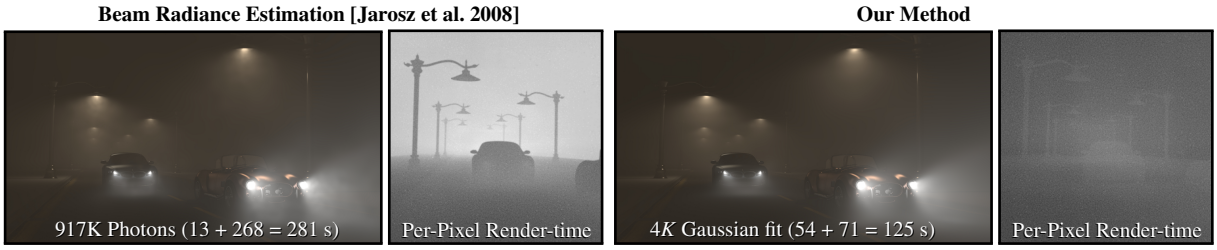


Figure 1: The beam radiance estimate (**left**) finds all photons along camera rays, which is a performance bottleneck for this CARS scene due to high volumetric depth complexity and many photons (917K). Our method (**right**) fits a hierarchical, anisotropic Gaussian mixture model to the photons and can render this scene faster, and with higher quality using only (4K) Gaussian components. The listed times denote the costs of the preprocessing stage (including photon tracing and hierarchy construction), as well as the final rendering stage, respectively.

Abstract

State-of-the-art density estimation methods for rendering participating media rely on a dense photon representation of the radiance distribution within a scene. A critical bottleneck of such kernel-based approaches is the excessive number of photons that are required in practice to resolve fine illumination details, while controlling the amount of noise. In this paper, we propose a parametric density estimation technique that represents radiance using a hierarchical Gaussian mixture. We efficiently obtain the coefficients of this mixture using a progressive and accelerated form of the Expectation–Maximization algorithm. After this step, we are able to create noise-free renderings of high-frequency illumination using only a few thousand Gaussian terms, where millions of photons are traditionally required. Temporal coherence is trivially supported within this framework, and the compact footprint is also useful in the context of real-time visualization. We demonstrate a hierarchical ray tracing-based implementation, as well as a fast splatting approach that can interactively render animated volume caustics.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Ray Tracing; I.6.8 [Simulation and Modeling]: Simulation—Monte Carlo

1. Introduction

Light interactions in participating media are responsible for many subtle but rich lighting effects in fire, water, smoke, clouds and crepuscular rays. Unfortunately, accurately rendering these effects is very costly. Cerezo et al. [CPP*05] provide a comprehensive overview of recent techniques.

Though methods such as (bidirectional) path tracing [LW93, VG94, LW96] or Metropolis light transport [PKK00] are general and provide unbiased solutions, it is costly to obtain high-quality, noise-free solutions except in the simplest scenes. The most successful approaches typically combine photon tracing with a biased Monte Carlo framework [JC98, WABG06, JZJ08].

Volumetric photon mapping [JC98] traces paths from light sources and stores "photons" at the vertices of these paths. A subsequent rendering pass approximates lighting using density estimation of the photons. The recent *beam radiance estimate* (BRE) [JZJ08] for photon mapping further accelerates this approach and improves the rendering quality by accounting for *all* photons that are located along a ray query.

However, even with such state-of-the-art optimizations, photon mapping suffers from a few remaining sources of inefficiency: 1) complex high-frequency lighting requires an excessive number of photons, which increases memory demands and leads to slow rendering times; 2) many photons may be processed which ultimately contribute very little to the final pixel color (due to subsequent attenuation or small projected pixel size); and 3) since photons are generated stochastically, the reconstructed density is prone to flickering unless many photons are used.

In this paper, we propose an algorithm that operates on a new representation of volumetric radiance to efficiently render images. This representation is compact and expressive, solving the aforementioned problems. Instead of storing individual photons and performing non-parametric density estimation [Sil86] during rendering, we progressively fit a small set of anisotropic Gaussians to the lighting distribution during photon tracing. Since each Gaussian is fit to the density of a large number of photons, we can accurately reconstruct complex lighting features during rendering using a small number (thousands) of Gaussians while providing better reconstruction than a large number (millions) of photons. Our framework trivially provides temporal coherence and, to handle the varying contributions of Gaussians to the rendered image (due to attenuation and projected pixel size), we propose a hierarchical approach which provides high-quality level-of-detail (LOD). Our compact hierarchical representation provides direct performance benefits which we demonstrate both in offline and interactive rendering.

2. Related Work

Photon Mapping. Schjøth and colleagues demonstrated the benefits of anisotropic density estimation [SFES07, SSO08] to obtain higher quality surface caustics with photon mapping. Hachisuka et al. [HOJ08] progressively refine radiance estimates at fixed measurement points to achieve convergent bias and variance in a view-dependent manner. We instead fit to the photon density by progressively refining both the positions and other parameters of an anisotropic Gaussian mixture, providing reduced variance and faster rendering from arbitrary viewpoints. Our work has a number of similarities to both hierarchical photon mapping representations [CB04, SJ09a] which provides LOD for final gather, and photon relaxation [SJ09b] which iteratively moves photon positions to reduce variance. By using Gaussian mixtures we obtain both of these benefits while drawing upon a set of principled and well-established fitting techniques.

Hierarchical and Point-based Rendering. Level-of-detail for light transport has been used to accelerate the Radiosity algorithm [HSA91, Sil95] and volume visualization [LH91]. Point-based [DYN04] and mesh-less [LZT*08] variants allow for more flexible geometric representations while Lightcuts [WFA*05, WABG06] provides hierarchical evaluation of lighting from many pointlights.

Our hierarchical rendering approach is related to hierarchical splatting of complex point-sampled geometry [RL00] and large stellar datasets [HE03, HLE04]; however, we use an anisotropic Gaussian representation. EWA splatting [ZPvBG02] uses Gaussians to provide improved, alias-free reconstruction for surface and volume visualization. While we share similar goals to these approaches, we operate within a parametric density estimation context and progressively fit the **positions** and other parameters of anisotropic Gaussians to the photon density.

Parameter Fitting. As opposed to previous global illumination techniques, which have relied primarily on non-parametric kernel density estimation such as the nearest neighbor method [Sil86], our technique leverages a large body of work on parametric density estimation and parameter fitting. Instead of performing the density estimation solely during rendering, we fit anisotropic Gaussians to the photon density during precomputation.

The use of Gaussians for rendering participating media is not new [ZHG*07, ZRL*08]; however, previous methods used *isotropic* Gaussians to approximate the heterogeneous *media density*, whereas we fit *anisotropic* Gaussians to the *radiance distribution*. We furthermore support hierarchical evaluation and leverage the well-established *Expectation–Maximization* [DLR*77] (EM) method specifically tailored for fitting anisotropic Gaussian mixtures.

While EM has seen recent use in image synthesis [TLQ*05, HSRG07, PJJ*11], it has a much richer history in fields such as machine learning and data mining where hierarchical techniques [GR05, GNN10] directly apply to LOD, and incremental techniques [NH98, HG09] provide parameter fitting within a limited memory footprint for streaming data. In these fields, however, large Gaussian mixtures are typically on the order of a few dozen to a few hundred components and their geometric quality is often only indirectly reproduced (e.g. color segmentation/classification). In contrast, to accurately fit volumetric lighting, we rely on much larger Gaussian mixtures (tens of thousands), and their fitted parameters directly influence the quality of the rendered image. To obtain the necessary quality and performance under these conditions we extend the *accelerated EM* method [VNV06].

3. Preliminaries

3.1. Radiative Transport in Participating Media

In the presence of participating media, the light incident at any point in the scene $\mathbf{x}$ (e.g. the camera) from a direction $\vec{\omega}$ (e.g. the direction through a pixel) as expressed by the radiative transport equation [Cha60] is the sum of two terms:

$$L(\mathbf{x},\vec{\omega}) = T_r(\mathbf{x}\leftrightarrow\mathbf{x}_b)\,L_s(\mathbf{x}_b,\vec{\omega}) + L_m(\mathbf{x},\vec{\omega},b), \qquad (1)$$

where the first term accounts for the radiance reflected off surfaces and the second term describes radiance scattered in the medium. Before reaching $\mathbf{x}$, the radiance from the nearest surface at a distance b, $\mathbf{x}_b = \mathbf{x} - b\vec{\omega}$, is reduced by the transmittance T_r which describes the percentage of light that propagates between two points (see Dutre et al. [DBB06] for details). We summarize our notation in Table 1.

In this paper we are primarily concerned with the media radiance which integrates the scattered light up to the nearest surface

$$L_m(\mathbf{x},\vec{\omega},b) = \int_0^b \int_{\Omega_{4\pi}} \sigma_s(\mathbf{x}_t)\,T_r(\mathbf{x}\leftrightarrow\mathbf{x}_t)\,f(\theta_t)\,L(\mathbf{x}_t,\vec{\omega}_t)\,d\vec{\omega}_t\,dt. \qquad (2)$$

Table 1: The notation used in this paper.

Symbol	Description
$\{\mathbf{x}_i\}_1^n$	Photon positions used in the EM fit
i	Index for photons
n	Total number of photons
$p(\mathbf{x})$	True probability density of photon positions
$p(\mathbf{x}\|\Theta)$	Probability density of GMM approximating $p(\mathbf{x})$
$g(\mathbf{x}\|\Theta_s)$	Gaussian density at $\mathbf{x}$ of GM component s
Θ_s	Weight, mean, and covariance matrix (w_s, μ_s, Σ_s) of component s
Θ	Set of all mixture model parameters $\{\Theta_s\}_1^k$
s	Index for GMM components
k	Number of GMM components
$\mathcal{L}(\Theta)$	Log-likelihood function defined in (6)
$\mathcal{F}(\Theta, q)$	Lower bound on the likelihood function, see (8)
$q_i(s)$	Responsibility of point i for GMM component s
$q_A(s)$	Responsibility of cell A for GMM component s
n_A	Number of points $\mathbf{x}_i$ that fall within A
$\langle..\rangle_A$	Average of the quantity .. w.r.t. the points $\mathbf{x}_i \in A$
$\bar{\Theta}_j$	Parameters $(\bar{w}_j, \bar{\mu}_j, \bar{\Sigma}_s)$ of GMM pyramid component j
j	Index into GMM pyramid nodes

Here σ_s is the scattering coefficient of the medium, f is the phase function, $\cos\theta_t = \vec{\omega}_t \cdot \vec{\omega}$, and $\mathbf{x}_t = \mathbf{x} - t\vec{\omega}$. This joint integral along the ray and over the sphere of directions, which recursively depends on the incident radiance (1), is what makes rendering participating media so costly.

3.2. Volumetric Photon Mapping and the BRE

Volumetric photon mapping [JC98] accelerates this computation using a two-stage procedure. First a photon map is populated by tracing paths from light sources and recording each scattering event as a "photon." Then, density estimation is performed over the photons to approximate the media radiance. To perform this computation efficiently, Jarosz et al. [JZJ08] proposed the *beam radiance estimate*, which first assigns a radius r_i to each of the n stored photons (using a pilot density estimate) and then approximates Equation (2) as

$$L_m(\mathbf{x}, \vec{\omega}, s) \approx \sum_{i=1}^{n} K_i(\mathbf{x}_t)\, T_r(\mathbf{x} \leftrightarrow \mathbf{x}_i)\, f(\theta_i)\, \Phi_i, \tag{3}$$

where $\mathbf{x}_i$, Φ_i and $\vec{\omega}_i$ are the position, power and incident direction of photon i respectively, $\cos\theta_i = \vec{\omega}_i \cdot \vec{\omega}$, and K_i is a 2D normalized kernel of radius r_i around photon i, evaluated at $\mathbf{x}_t = \mathbf{x} + ((\mathbf{x}_i - \mathbf{x}) \cdot \vec{\omega})\vec{\omega}$. We always use i to index photons. Note that Jarosz et al. [JZJ08] included an additional σ_s term. We account for this during photon instead tracing to remain consistent with other sources [JC98, JNSJ11].

Unfortunately, there are a few common situations where the BRE is the bottleneck in rendering. Firstly, complex, high-frequency lighting may require an excessive number of photons, which increases storage costs and also slows down the beam query during rendering. Secondly, since the BRE always finds *all* photons that intersect the query ray, if the scene has a large extent, many photons may be processed which ultimately contribute very little to the final pixel color (due to subsequent attenuation or projected pixel size). Figure 1 visualizes the time spent performing the BRE per pixel, which shows this deficiency. Our approach addresses both of these problems by expressing the distribution of photons in the media using a hierarchical, anisotropic Gaussian mixture.

3.3. Gaussian Mixtures

We briefly review Gaussian mixture models (GMMs), maximum likelihood estimation, and Expectation–Maximization in terms of our rendering problem.

Given a collection of photons $\mathbf{X} = \{\mathbf{x}_i\}_1^n$, volumetric photon mapping corresponds to estimating the underlying probability distribution $p(\mathbf{x})$. In general, $p(\mathbf{x})$ could be any distribution obtainable using photon tracing. Photon mapping estimates this distribution using kernel density estimation, whereas our approach relies on a GMM. More precisely, we assume that the underlying probability density can be modeled using a weighted sum of k probability distributions,

$$p(\mathbf{x}) \approx p(\mathbf{x}|\Theta) = \sum_{s=1}^{k} w_s\, g(\mathbf{x}|\Theta_s), \tag{4}$$

where Θ_s denotes the parameters for distribution s and $\Theta = \{\Theta_s\}_1^k$ defines the model parameters for the entire mixture. Each individual component g is a normalized multivariate Gaussian

$$g(\mathbf{x}|\Theta_s) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\mu_s)^\intercal \Sigma_s^{-1}(\mathbf{x}-\mu_s)\right)}{(2\pi)^{3/2}\sqrt{|\Sigma_s|}}, \tag{5}$$

where $\Theta_s = (\mu_s, \Sigma_s, w_s)$ with mean μ_s, covariance matrix Σ_s, and weighted by w_s. We use $|\cdot|$ to denote the determinant and always use s to index into the Gaussian components.

Since photons are not actually generated using a Gaussian process, (4) is an approximation, but by increasing k a Gaussian mixture can model any photon distribution with arbitrary accuracy.

3.4. Maximum Likelihood Estimation

With the assumptions above, our problem is reduced to finding the values of the model parameters Θ_s for each of the k Gaussians, which, when plugged into the model (4), are *most likely* to generate the photons. More formally, such *maximum likelihood* estimators seek the parameters Θ that maximize of the associated data log-likelihood

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} \log\left(p(\mathbf{x}_i|\Theta)\right). \tag{6}$$

Note that the actual photon density $p(\mathbf{x})$ is not an explicit part of this formulation; instead, it only enters the problem statement via evaluation of the GMM (4) at the photon positions.

3.5. Expectation Maximization

The popular EM algorithm is an iterative technique for finding such a maximum likelihood parameter estimate. Instead of maximizing $\mathcal{L}$ directly (which is usually intractable), EM starts with an initial guess of the parameters Θ and modifies them so that a lower bound $\mathcal{F} \leq \mathcal{L}$ is strictly increased in every one of its iterations. The bound $\mathcal{F}$ is chosen so that its local maxima coincide with those of $\mathcal{L}$; hence, it is guaranteed

that the algorithm eventually converges to a local maximum of $\mathcal{L}$.

During the optimization, EM makes use of a series of discrete distributions $q_i(s) : \{1,\dots,k\} \to [0,1]$ (one for each photon $\mathbf{x}_i$), which specify the associated photon's "responsibility" towards each GMM component. The lower bound $\mathcal{F}(\Theta, q)$ can then be expressed as a function of the parameters Θ, as well as the auxiliary distributions q_i. An intuitive and general interpretation of the EM-type algorithm can then be formulated as [NH98]:

E-Step: Given Θ, find q that maximizes $\mathcal{F}(\Theta, q)$, then
M-Step: Given q, find Θ that maximizes $\mathcal{F}(\Theta, q)$.

3.6. Accelerated EM

In this paper, we make use of a technique called *accelerated EM* [VNV06]. The insight of this approach is to assign the same distribution $q_i(s) = q_A(s)$ to all photons $\mathbf{x}_i$ that fall into the same cell A of a spatial partition $\mathcal{P}$ of $\mathbb{R}^3$. $\mathcal{P}$ is simply a subdivision of space where any point lies in exactly one cell, e.g. the collection of cells of a Voronoi diagram, or a cut through a kd-tree or octree. This allows rephrasing the EM algorithm purely in terms of operations over cells (as opposed to photons).

For the E-step, Verbeek et al. provide a provably-optimal assignment of the responsibilities:

$$q_A(s) = \frac{w_s \exp\langle \log g(\mathbf{x}|\Theta_s)\rangle_A}{\sum_{a=1}^{k} w_a \exp\langle \log g(\mathbf{x}|\Theta_a)\rangle_A} \tag{7}$$

where $\langle f(\mathbf{x})\rangle_A$ denotes the average of a function f over all photons that fall within cell A. They also give an expression for the lower bound $\mathcal{F}$ as a sum over the partition's cells:

$$\mathcal{F}(\Theta, q) = \sum_{A\in\mathcal{P}} n_A \sum_{s=1}^{k} q_A(s) \left[\log \frac{p(s)}{q_A(s)} + \langle \log g(\mathbf{x}|\Theta_s)\rangle_A\right]. \tag{8}$$

The average log-densities in (7) and (8) can be readily computed in closed-form for Gaussian mixtures [VNV06]. For this, we only need to store the average photon position $\langle \mathbf{x}\rangle_A$ and outer product $\langle \mathbf{x}\mathbf{x}^T\rangle_A$ within each cell.

The accelerated M-step computes the parameters Θ_s as weighted averages over cell statistics:

$$\Theta_s = (w_s, \mu_s, \Sigma_s) = \frac{1}{n} \sum_{A\in\mathcal{P}} n_A q_A(s) \left(1, \frac{\langle \mathbf{x}\rangle_A}{w_s}, \frac{\langle \mathbf{x}\mathbf{x}^\mathsf{T}\rangle_A - \mu_s\mu_s^\mathsf{T}}{w_s}\right), \tag{9}$$

where n denotes the total number of photons and n_A specifies the number of photons that fall within cell A. Note that the terms of Θ_s need to be computed in sequence to satisfy interdependencies. A naïve accelerated EM implementation is shown in Algorithm 1.

Algorithm 1 ACCELERATEDEM-NAÏVE($\Theta^{(0)}, \mathcal{P}$)

```
1  z = 0                                   // iteration counter
2  repeat
3      Compute the distribution q_A(s) for each A ∈ P using (7)
4      Compute Θ^(z+1) using Equation (9)
5      z = z + 1
6  until |F(Θ^(z), q^(z)) / F(Θ^(z−1), q^(z−1)) − 1| < ε
```

This formulation has several advantages which we will exploit for photon mapping:

- **Compact representation**: The photons $\mathbf{x}_i$ can be discarded, and only a few sufficient statistics of their behavior within each cell $A \in \mathcal{P}$ must be stored.
- **Fast iterations**: A small number of statistics succinctly summarize the photons within a cell. Good results can be obtained with comparatively few cells, where many photons would be needed by non-accelerated EM; this allows for shorter running times.
- **Multiresolution fitting**: The partition need not necessarily stay the same over the course of the algorithm. Reminiscent of multigrid methods, we can start with a relatively rough partition and refine it later on.

4. Overview of our Algorithm

In the remainder of this paper we apply the concepts of accelerated EM to the problem of rendering participating media using photon tracing. At a high-level, our approach replaces individual photons in the BRE with multivariate Gaussians which are fit to the photons using EM. Since a Gaussian is fit to a large collection of photons, each term is inherently less noisy and more expressive than an individual photon. This provides high quality reconstruction using a relatively low number of terms, which results in faster rendering at the expense of a slightly longer view-independent precomputation stage.

Our method extends accelerated EM from Section 3.6 in three simple but important ways. We firstly introduce sparsity into the algorithm to obtain asymptotically faster performance. Secondly, we incorporate progressive refinement by incrementally shooting photons, updating statistics in the partitions, and refining the EM fit. This elegantly allows memory-bounded fitting of extremely large collections of photons while simultaneously providing an automatic stopping criterion for photon tracing. We call this process *progressive EM*. We believe this extension could also find applicability in data mining for incremental fitting of streaming data. Lastly, we build a full GMM pyramid which allows us to incorporate hierarchical LOD during rendering.

This algorithm is divided into four key steps (see Figure 2):

1. **Photon tracing**: Trace photons within the scene (just like in photon mapping), storing the resulting statistics in the partition $\mathcal{P}$, then discard the photons.

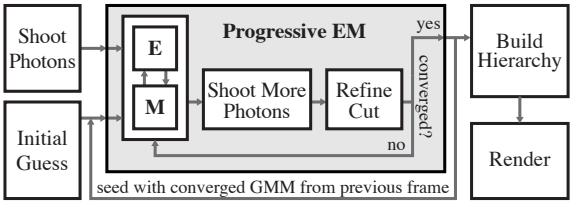


Figure 2: The program flow of our approach.

(a) Target density (b) Initial cut (c) Initial GMM (d) Final cut (e) Final GMM

Figure 3: As a simple two-dimensional application of our method, we fit a 1024-term GMM to an image-based target density function (a). Our method starts with a coarse initial cut (b) and GMM parameters (c) and refines them until convergence (d and e).

2. **Progressive GMM fitting**: Apply progressive EM (Section 5) to the statistics in $\mathcal{P}$. Should the stored statistics be too noisy, repeat step 1 by shooting more photons, updating statistics, and continue refining the GMM fit.
3. **Level-of-Detail:** Construct a full LOD pyramid of GMMs with decreasingly many components (Section 6).
4. **Hierarchical Rendering:** For each ray, hierarchically prune the LOD pyramid to the set of Gaussians with sufficient contribution according to a conservative error heuristic (Section 7).

We focus on practical considerations for implementing these steps in the following sections and omit details for the first stage since it is identical to standard photon mapping.

5. Progressive GMM Fitting

Our progressive EM algorithm is outlined in Algorithm 2. The core of the approach is based on accelerated EM, but we introduce sparsity to lines 4–5, progressively update the statistics (line 8), and use a more efficient cut refinement procedure (line 10).

Algorithm 2 PROGRESSIVEEM($\Theta^{(0)}, \mathcal{P}$)

```
1  z = 0                                   // iteration counter
2  repeat
3      repeat
4          Compute the distr. q_A(s) for all A ∈ P using (7)
5          Compute Θ^(z+1) using Equation (9)
6          z = z + 1
7      until |F(Θ^(z), q^(z)) / F(Θ^(z−1), q^(z−1)) − 1| < ε
8      if Equation (10) indicates an unsatisfactory fit
9          Shoot more photons and update statistics
10     Expand cut along 50% "best" nodes according to F (8)
11 until the relative change in the expanded F drops < ε
```

5.1. Partition Construction

Our algorithm assumes the availability of a partition $\mathcal{P}$. To obtain a spatial partition of $\mathbb{R}^3$, we follow an approach similar to the one taken by Verbeek et al.: we create an entire family of possible partitions by constructing a balanced kd-tree over the photon positions, where each leaf and inner node is augmented with the aforementioned sufficient statistics; this can be done in $\mathcal{O}(n \log n)$ time. Any cut through the tree then forms a valid partition. To ensure that the tree fits in memory, we use a moderate number of photons ($n = 128 \cdot k$) and create a leaf node once the number of photons in a subtree drops below 5. After this step, all memory associated with the photons can be released, and only the hierarchy remains.

We initialize the cut to the level of the kd-tree where the number of cells equals eight times the number of GMM components (we found this to be a good compromise in terms of the total running time).

5.2. Progressive Update and Cut Refinement

After running E- and M-steps until convergence, we check the effective number of photons that contribute to GMM component s:

$$n_{\text{eff}}(s) = \sum_{A \in \mathcal{P}} n_A q_A(s). \tag{10}$$

If any Gaussian is fit to fewer than 64 photons, we deem these statistics to be unreliable, so we shoot more photons, and progressively update the statistics. Shooting more photons reduces the variance of the statistics, converging to the proper average of these quantities within each cell in the limit.

After this step, we use a greedy criterion to push down the partition cut. This entails removing nodes and replacing them by their children in the kd-tree. Verbeek et al. proposed expanding the cut by one node in each outer iteration, but we found that this results in slow convergence. Instead, we determine the most suitable set of nodes by computing the increase in $\mathcal{F}$ (8) for each possible replacement and execute the top 50% of them. We repeat this entire procedure until fitting parameters converge. Figure 3 illustrates this procedure on an image-based density function. The progressive EM iterations and cut refinement, as well as the computation of Equation 10, can all be done efficiently using ideas discussed in the next section.

5.3. Exploiting Sparsity

Unfortunately, the complexity of this algorithm renders it impractical for large-scale problem instances, which are the main focus of this paper. In particular, although the computation of (7) and (9) no longer depends on the number of photons, the cost is linear in the product of $|\mathcal{P}|$ and k. To obtain a reasonable fit, the final cut will usually satisfy

$|\mathcal{P}| = c \cdot k$, for some constant $c > 1$, hence the running time is in $\mathcal{O}(k^2)$.

To work around the complexity issue, our fitting pipeline exploits the inherent sparsity of this problem: due to the exponential falloff in g, it is reasonable to expect that a cell $A \in \mathcal{P}$ will excert little influence on a distant Gaussian component s. In the context of accelerated EM, this means that the value of $q_A(s)$ will be negligible. We may thus want to skip such cell-GMM component pairs in the calculation of (9) or (10).

There is one caveat to the previous idea: suppose that a cell $A \in \mathcal{P}$ is located relatively far away from *all* of the GMM components. Although the numerator in $q_A(s)$ (7) will be very small in this case, $q_A(s)$ will nonetheless exert a significant influence on distant GMM components due to the normalization term in its denominator. This means that a pruning criterion cannot be based on spatial distance alone.

Our method therefore proceeds in two stages: first, we determine the normalization term in the denominator of $q_A(s)$ (7) for each cell $A \in \mathcal{P}$. Following this, we recompute the GMM parameters Θ and effective photon counts $n_{\text{eff}}(s)$ using Equation (9) and (10). All of these steps exploit sparsity and obey a user-specified relative error tolerance τ, such that the algorithm becomes faster and more approximate with higher τ and degenerates to the $\mathcal{O}(n^2)$ approach when $\tau = 0$. Specifically, we loosely bound q_A as a function of spatial distance and drop Gaussian-cell pairs with the help of a bounding volume hierarchy when this does not cause the result to exceed the specified error bound. All our results use $\tau = 10^{-4}$.

5.4. Initial Guess and Temporal Coherence

EM requires an initial guess which is iteratively refined. Since EM will, in general, only converge to a local maximum of $\mathcal{L}$, the starting parameters are an important factor in the final quality of the EM fit.

For still images we obtain an initial guess by extracting a suitably-sized horizontal cut through the partition kd-tree. Each cell in this cut already stores the sufficient statistics that are needed to analytically fit a Gaussian to its contents, hence this step takes negligible time.

For animations we could apply the same procedure for each frame independently. However, like any photon mapping method, the stochastic positions of photons can introduce temporal flicking. We can achieve significantly better quality by seeding EM with the converged GMM parameters from the previous frame (a similar idea was used by Zhou et al. [ZRL*08] for temporally coherent non-linear optimization). This optional step does introduce dependence between frames, but we found it to be worthwhile since it completely eliminated temporal flicker in our results. Furthermore, since the previous frame is often a remarkably good match to the current lighting, this significantly reduces the computation time needed for fitting (for instance, fitting the BUMPYSPHERE animation is 4.35× faster compared to frame-independent initialization).

6. Level-of-Detail Construction

To render the GMM at various levels of detail, we wish to construct a hierarchical representation of the Gaussian mixture. In our approach, the leaves are the components of our GMM, interior levels approximate the GMM using decreasing numbers of Gaussian components, up to the root which expresses the entire distribution as a single Gaussian. Note that we cannot simply compute a multi-resolution representation (where each level of the tree is an independent GMM with 2^l components). Since we will dynamically select subsets of this tree for rendering, there needs to be a proper nesting between the levels of the hierarchy such that a node is a representative for its entire subtree.

To accomplish this efficiently, we apply a modified version of Walter et al.'s heap-based agglomerative clustering algorithm [WBKP08] as shown in Algorithm 3. We denote nodes of the GMM pyramid as $\bar{\Theta}$ and index a specific node as $\bar{\Theta}_j$. We initialize the leaves of the pyramid (line 1) to the Gaussian mixture, $\{\Theta_s\}_1^k$, constructed in Section 5. To form the next higher level, we start with a heap of all minimum Gaussian distances in the current level (lines 6–8) and then merge (and remove) the two "most similar" Gaussians until no Gaussians are left in the heap (lines 9–15). Repeating this entire process $\log_2 k$ times creates a full binary tree.

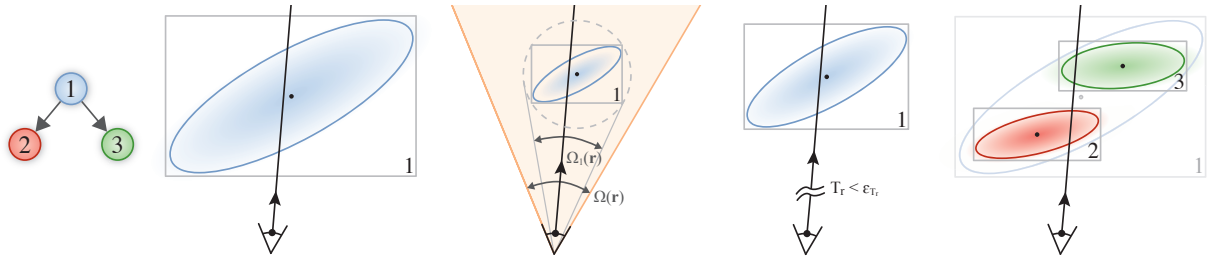
Algorithm 3 CONSTRUCTGAUSSIANPYRAMID($\Theta = \{\Theta_s\}_1^k$)

```
1  Θ̄ = Θ                    // construct tree starting with leaf nodes
2  level = log2 k
3  while level > 0          // loop until we reach the root
4      Θ̄' = ∅               // initialize next level to empty set
5      H = ∅                // initialize min-heap to empty set
6      for each Gaussian j1 ∈ Θ̄
7          Find closest Gaussian j2 to j1 according to (11).
8          Insert edge (j1, j2) into H.
9      while H ≠ ∅
10         Pop an edge e from the heap
11         if neither endpoint of e has been previously merged
12             Merge the two Gaussians of e according to
               Equations (12–14) and insert into Θ̄'
13             for all vertices j1 connected to either end of e
14                 Find the closest Gaussian j2 to j1.
15                 Insert edge (j1, j2) into H.
16     Θ̄ = Θ̄'
17     level = level − 1
18 return Θ̄
```

We calculate the distance between Gaussians using a symmetric dissimilarity function

$$d(j_1, j_2) = w_{j_1} \cdot w_{j_2} \cdot \min(D_{\text{KL}}(\bar{\Theta}_{j_1} \| \bar{\Theta}_{j_2}), D_{\text{KL}}(\bar{\Theta}_{j_2} \| \bar{\Theta}_{j_1})), \quad (11)$$

where D_{KL} is the Kullback-Leibler divergence between component j_1 and j_2, and takes on a simple analytical form in the case of multivariate Gaussians [GNN10]. Kullback-Leibler divergence is a fundamental statistical measure of relative



(a) Hierarchy used in (b) - (e) (b) Intersect bounding box (c) Criterion 1: small projected size (d) Criterion 2: low transmittance (e) Otherwise: recurse

Figure 4: We (a) traverse our hierarchy of Gaussians from top to bottom, (b) intersecting the current node's AABB. We then (c) test the solid angle of the representative and also check the extinction (d) before traversing down the tree.

entropy between two distributions and has been used extensively for clustering Gaussian distributions [GR05, GNN10].

To merge Gaussians $\bar{\Theta}_{j_1}$ and $\bar{\Theta}_{j_2}$ into $\bar{\Theta}_j$, we analytically fit parameters as follows [GR05]:

$$\bar{w}_j = \bar{w}_{j_1} + \bar{w}_{j_2}, \tag{12}$$

$$\bar{\mu}_j = \bar{w}_j^{-1}\left(\bar{w}_{j_1}\bar{\mu}_{j_1} + \bar{w}_{j_2}\bar{\mu}_{j_2}\right), \tag{13}$$

$$\bar{\Sigma}_j = \bar{w}_j^{-1} \sum_{k\in\{j_1,j_2\}} \bar{w}_k \left(\bar{\Sigma}_k + (\bar{\mu}_k - \bar{\mu}_j)(\bar{\mu}_k - \bar{\mu}_j)^\mathsf{T}\right). \tag{14}$$

This procedure eventually forms a full binary tree that we use for hierarchical pruning during rendering.

7. Rendering

During rendering, for a given query ray passing through the medium, we are interested in approximating the medium radiance (2) using our GMM. If we ignored the hierarchy constructed in the last section, we could directly adapt the beam radiance estimate (3) by replacing photons $\mathbf{x}_i$ and kernels K_i with the components $g(\mathbf{x}|\Theta_s)$ of our Gaussian mixture.

All the Gaussian terms, in theory, contribute to the media radiance along a query ray, but in reality some components may contribute very little (due to distance to the ray, significant attenuation, or small projected solid angle). The hierarchy constructed in the last section allows us to efficiently incorporate LOD. The main idea behind our approach is to traverse the GMM pyramid top-down and hierarchically prune the traversal when the interior nodes are deemed a reasonable approximation for their subtree. To facilitate this hierarchical pruning, we augment the GMM pyramid with an axis-aligned bounding-box (AABB) hierarchy (BBH).

7.1. Bounding-Box Hierarchy Construction

We spatially bound each Gaussian by computing the distance at which its remaining energy is a small percentage of the total energy of the Gaussian mixture. This truncates each anisotropic Gaussian to an ellipsoid. We initialize the bounding box of each node to tightly fit its truncated Gaussian. We then complete the BBH by propagating the bounding-box information up the tree: each node's bounding box is unioned with the bounding box of its two child nodes. This construction can be performed efficiently in $\mathcal{O}(k \log k)$ using two sweeps of the Gaussian pyramid.

7.2. Hierarchical Traversal

For fast hierarchical rendering, we prune entire subtrees of the GMM pyramid during ray traversal by accounting for the expected contribution of each Gaussian to the rendered image. For a given query ray $\mathbf{r}$, we start at the root. At each node we decide between three possible outcomes: 1) the entire subtree can be skipped, 2) the subtree can be well approximated by adding the contribution of the current node to the BRE, or 3) we need to descend further.

We first intersect the ray with the bounding box of the node. If there is no hit, we skip the entire subtree. Otherwise, we compute the projected solid angle of the node's bounding sphere $\Omega_j(\mathbf{r})$ and compare it to the solid angle of a pixel of the ray $\Omega(\mathbf{r})$. If $\Omega_j(\mathbf{r}) < \varepsilon_\Omega\,\Omega(\mathbf{r})$, for a user-specified threshold ε_Ω, we evaluate the contribution of the node's Gaussian. If this test fails, we additionally compute the transmittance between the origin of the ray and the first hit point with the bounding box, $T_r(\mathbf{r},\Theta_j)$. If this is less than a user-specified constant, ε_{T_r}, we evaluate the node's Gaussian. Only if all these tests fail do we descend further down the BBH. These cases are illustrated in Figure 4 and outlined in Algorithm 4.

Algorithm 4 HIERARCHICALBRE$(\mathbf{r},\Theta_j)$

```
if r does not intersect AABB(Θ_j)
    return 0
if (Ω_j(r) < ε_Ω Ω(r)) or (T_r(r, Θ_j) < ε_Tr)
    return Gaussian contribution using Equation (15)
return HIERARCHICALBRE(r, Θ_2j) +
       HIERARCHICALBRE(r, Θ_2j+1)
```

To evaluate the contribution of a Gaussian, we need to integrate it along $\mathbf{r}(t) = \mathbf{x} + t\vec{\omega}$ while accounting for transmittance.

$$L_m^j(\mathbf{x},\vec{\omega},b) = \frac{1}{4\pi}\,\bar{w}_j \int_0^b g(\mathbf{r}(t)|\bar{\Theta}_j)\,T_r(\mathbf{x}\leftrightarrow\mathbf{r}(t))\,dt. \tag{15}$$

This corresponds to a single summand in (3) for isotropic scattering (we discuss anisotropic scattering in Section 9). As in Equation (3), neither σ_s nor the albedo are used here because we account for these terms during photon tracing.

For homogeneous media, the integral in Equation (15) ac-

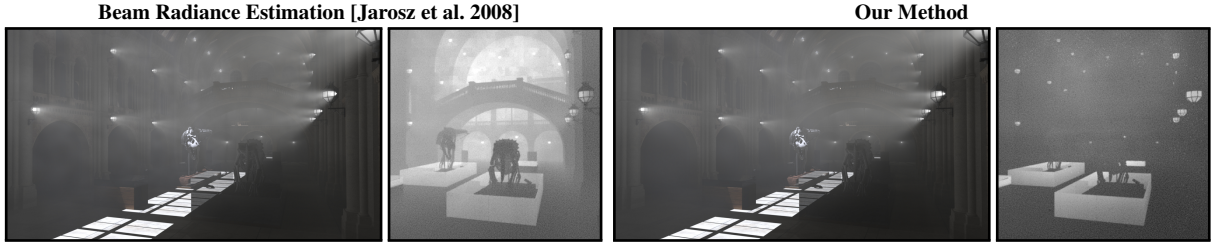


Figure 5: In the MUSEUM scene, the high volumetric depth complexity and large number of photons ($3.1M$) makes the BRE (**left**) inefficient. Our method addresses these problems by using a hierarchical, anisotropic Gaussian mixture with only $16K$ Gaussian components.

cepts the closed-form solution:

$$\int_a^b g(\mathbf{r}(t)|\bar{\Theta}_j)\, e^{-\sigma_t t} dt = C_0 \left[\operatorname{erf}\left(\frac{C_3 + 2C_2 b}{2\sqrt{C_2}}\right) - \operatorname{erf}\left(\frac{C_3 + 2C_2 a}{2\sqrt{C_2}}\right)\right], \tag{16}$$

where the constants C_0 through C_3 are:

$$C_0 = \frac{\exp\left(\frac{C_3^2}{4C_2} - C_1\right)}{(2\pi)^{\frac{3}{2}}\sqrt{|\bar{\Sigma}_j|}}, \quad C_1 = \frac{1}{2}(\mathbf{x} - \bar{\mu}_j)^\intercal \bar{\Sigma}_j^{-1}(\mathbf{x} - \bar{\mu}_j) - \sigma_t b,$$

$$C_2 = \frac{1}{2}\vec{\omega}^\intercal \bar{\Sigma}_j^{-1}\vec{\omega}, \qquad C_3 = \vec{\omega}^\intercal \bar{\Sigma}_j^{-1}(\mathbf{x} - \bar{\mu}_j) + \sigma_t. \tag{17}$$

Though we don't currently support this in our implementation, for heterogeneous media we could take an approach similar to the BRE and compute this integral numerically using ray marching.

8. Implementation and Results

We implemented our hierarchical GMM rendering approach in a Monte Carlo ray tracer and also implemented the BRE in the same system to facilitate comparison. All timings are captured on a dual-core Intel Core i7 2.80GHz CPU with 6 GB memory and an NVIDIA GeForce GT240.

8.1. GPU Splatting

In addition to the general hierarchical ray tracing solution described in Section 7.2, we have also implemented a GPU splatting approach which allows us to interactively preview arbitrary levels of the GMM mixture.

Each Gaussian component is rendered as an elliptical splat corresponding to the screen-space projection of the truncated Gaussian ellipsoid. Each splat is rendered as a triangle strip and Equation (15) is evaluated in a fragment shader with additive blending. This gives correct results for homogeneous media. While our GPU implementation ignores surface interactions, it nonetheless provides a useful tool to interactively preview the media radiance (refer to the video).

8.2. Results

Parameters and rendering statistics for all scenes are listed in Table 2. Note that the preprocessing costs listed for the BRE and progressive EM methods include the time spent tracing photons.

In Figure 1 and 5 we highlight the problems that our approach addresses. Both the CARS and MUSEUM scenes contains several light sources and have a relatively large depth-extent. We show a rendering with the BRE, and with our approach, as well as visualizations of the rendering time spent in each pixel. With the BRE, regions of the image that contain large concentrations of photons take longer to render, even if these photons contribute little to the final pixel color. Our hierarchical rendering technique solves this problem by detecting and pruning such low contribution subtrees in the GMM pyramid. The resulting images are higher quality and render faster. Furthermore, since our pre-computation is view-independent, fly-through animations render very quickly once the Gaussian mixture is constructed.

Figure 6 shows a progression of renderings of the BUMPYSPHERE scene with increasing numbers of Gaussian components. Though our compact GMM representation uses only a few thousand Gaussian components (ranging from 1K to 64K) we are able to faithfully capture high-frequency caustic structures because we progressively fit to a large number of photons (up to 18 million). We also show BRE results which use all the available photons during rendering. Each BRE result takes longer to render than the corresponding GMM rendering (between 7.8 and 9.2× slower for the rendering pass, and 1.2 to 6.0× slower including the precomputation stage). Since the BRE uses isotropic density estimation, even with all the photons it produces both noisier and blurrier results. Our anisotropic GMM fitting procedure acts like an intelligent noise-reduction and sharpening filter that automatically finds structures in the underlying density in a way not possible with on-the-fly density estimation.

Figure 7 shows the OCEAN and SPHERECAUSTIC scenes depicting detailed volume caustics. These high-frequency structures are difficult to reconstruct even when using over $4M$ isotropic photon points with the BRE, while our anisotropic fitting process produces slightly better results 1.2× and 1.6× faster using only $16K$ Gaussian components.

The accompanying video shows animations for the BUMPYSPHERE and OCEAN scenes using both the offline renderer and the GPU splatting approach. The videos also highlight the benefit of our temporal coherence as compared

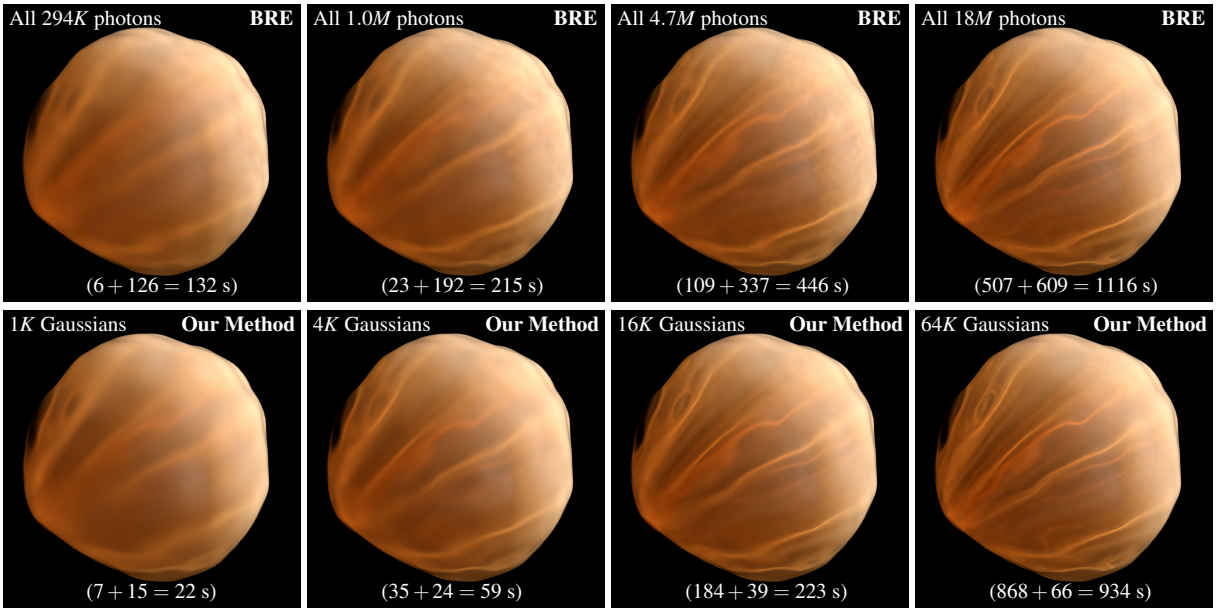


Figure 6: BUMPYSPHERE renderings using our method (**bottom**) and corresponding BRE renderings (**top**) using all photons in the Gaussian fit. As before, the listed times correspond to the preprocessing and rendering stages, respectively.

to the BRE. When animated, we also obtain increased performance since the previous frame is a closer match to the converged solution and less EM iterations are required. The total fitting time for the BUMPYSPHERE animation is 4.35× lower using the temporally coherent method.

9. Discussion and Future Work

Anisotropic Media. A significant limitation of our current implementation is that we only handle isotropic media. We believe our general methodology, however, could be naturally extended to handle this case by additionally storing directional information during fitting. One option is to perform the fitting procedure entirely in a 5D spatio-directional domain where each component is the tensor product of a spatial Gaussian and a von Mises-Fischer distribution. Another possibility would be to treat the two domains separately and fit von Mises-Fischer distributions to the directional statistics under the support of each 3D Gaussian in a similar fashion to previous multi-resolution methods for surface reflectance [TLQ*05, HSRG07].

Performance, Quality, and Scalability. Our render pass is considerably faster than the BRE due to the reduced number of terms and hierarchical pruning during traversal. Yet, we are able to obtain higher quality results with such few terms since our Gaussians are anisotropic and inherently less noisy than individual photons. Even including precomputation, our total time to render is faster than the BRE for the scenes we tried. Furthermore, since our fitting is view-independent, we provide an even greater performance benefit for walk-throughs of static or pre-scripted lighting.

GMMs are also an attractive low-memory representation for volumetric lighting: For instance, a 4096-term GMM requires only 240 kilobytes of storage.

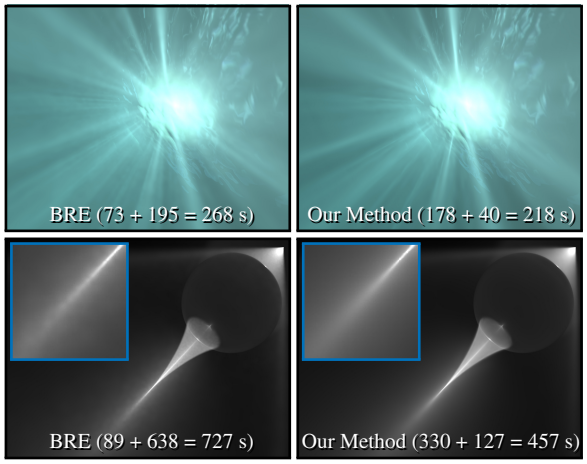


Figure 7: The OCEAN and SPHERECAUSTIC scenes comparing the BRE with 2*M* photons to our method with 16*K* Gaussians.

Though our pre-computation stage is slightly slower, the statistics in Table 2 indicate that progressive EM is actually quite competitive (in speed and scalability) with the precomputation needed for the BRE. Nonetheless, our fitting does incur a performance penalty which may not be warranted in situations where other aspects of our algorithm are not beneficial (e.g. in very low-frequency illumination, or scenes with a small depth extent).

Fitting to Other Representations. Recently, Jarosz et al. [JNSJ11] proposed the photon beams method which performs density estimation over photon path segments instead

Table 2: Parameters and performance statistics for the scenes used in this paper. For BRE the preprocess includes photon tracing, kd-tree and BBH construction, and radius estimation. For our method this includes progressive photon shooting, GMM fitting, and hierarchy construction.

					BRE [Jarosz et al. 2008]			Our Method		
Scene	k	n	$\lvert\mathcal{P}\rvert$	**EM iter.**	**Preprocess**	**Render**	**Total**	**Preprocess**	**Render**	**Total**
CARS	4K	917,504	87,615	12	13.48 s	268.20 s	281.69 s	54.43 s	71.21 s (**3.8**×)	125.64 s (**2.2**×)
MUSEUM	16K	3,670,016	381,602	10	133.90 s	343.95 s	477.89 s	225.73 s	100.39 s (**3.4**×)	326.12 s (**1.7**×)
BUMPYSPHERE	1K	294,912	30,264	19	5.73 s	126.35 s	132.08 s	6.81 s	15.19 s (**8.3**×)	22.00 s (**6.0**×)
BUMPYSPHERE	4K	1,085,621	121,910	19	23.46 s	192.30 s	215.76 s	35.13 s	24.52 s (**7.8**×)	59.65 s (**3.6**×)
BUMPYSPHERE	16K	4,718,592	489,602	20	109.30 s	337.42 s	446.72 s	184.06 s	39.35 s (**8.6**×)	223.41 s (**2.0**×)
BUMPYSPHERE	64K	18,874,368	1,961,891	19	507.30 s	609.25 s	1116.55 s	868.48 s	66.03 s (**9.2**×)	934.51 s (**1.2**×)
OCEAN	16K	4,718,592	485,339	18	73.47 s	195.46 s	268.93 s	178.25 s	40.67 s (**4.8**×)	218.92 s (**1.2**×)
SPHERECAUSTIC	16K	4,194,304	458,366	11	89.55 s	638.39 s	727.90 s	329.83 s	127.30 s (**5.0**×)	457.13 s (**1.6**×)

of photon points. They showed that this results in improved rendering quality compared to the BRE. Their method shares similar goals to ours (more compact representation, and higher quality reconstruction); however, their benefits is actually orthogonal to ours. Our Gaussian fitting pipeline could be applied in the context of photon beams as well as photon points. A straightforward combination could update statistics in all partition cells intersected by a photon beam. This could significantly improve the quality of the statistics (and the EM fit) compared to photon points. We believe that in homogeneous media the statistical contributions of a beam could be evaluated analytically and plan to investigate this in future work. Another, more ambitious, avenue would be to develop a fitting procedure analogous to EM which operates more explicitly on photon beams, by clustering into higher-dimensional density distributions such as blurred 2D line segments.

Convergence and Automatic Termination. Though the name of our progressive EM algorithm is inspired by progressive photon mapping (PPM) [HOJ08] there are important differences. Our method progressively fits a Gaussian mixture to photon statistics whereas PPM progressively refines radiance estimates by updating statistics at fixed measurement points. Our Gaussian components could be roughly interpreted as "measurement points" which not only adapt in intensity, but also location, to the underlying photon density. However, whereas PPM provides guarantees for convergence of both bias and noise, our solution eliminates noise in the limit but is always biased since we use a fixed number of Gaussian terms. An interesting possibility for future work would be to progressively split Gaussian terms when sufficient statistics indicate a single Gaussian cannot represent the underlying distribution, or to automatically deduce the required number of Gaussian terms.

Automatic termination by specifying a desired error in the image has been investigated previously in the context of PPM [HJJ10]. While our progressive EM algorithm also provides an automatic stopping criterion for shooting photons, this tolerance does not have an intuitive relation to the error in the image, but rather describes the convergence of the GMM fit to the underlying density distribution. Error is fairly well understood in the context of kernel-based density estimation used in photon mapping. A more rigorous investigation of the error and bias introduced by our method is an interesting avenue of future work.

10. Conclusion

We have presented a progressive, hierarchical method for rendering participating media. Our approach can be viewed as a variant of volumetric photon mapping which replaces the usual on-the-fly non-parametric density estimate with a parametric density model based on a Gaussian mixture fit to the photons. The fitting is done using a sparse and progressive form of the accelerated EM algorithm. We construct a hierarchical representation which is used to provide estimates of scattered radiance at appropriate scales during rendering. Our approach shows improved quality and temporal coherence as compared to the BRE and also supports interactive preview using splatting.

Acknowledgments. We thank Steve Marschner, Peter-Pike Sloan, and Derek Nowrouzezahrai for helpful discussions and comments. The MUSEUM scene is courtesy of Alvaro Luna Bautista and Joel Anderson; the BUMPYSPHERE scene is courtesy of Bruce Walter.

References

[CB04] CHRISTENSEN P. H., BATALI D.: An irradiance atlas for global illumination in complex production scenes. In *Rendering Techniques* (June 2004), pp. 133–142. 2

[Cha60] CHANDRASEKHAR S.: *Radiative Transfer*. Dover Publications, New York, 1960. 2

[CPP*05] CEREZO E., PÉREZ F., PUEYO X., SERON F. J., SILLION F. X.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (2005). 1

[DBB06] DUTRÉ P., BALA K., BEKAERT P.: *Advanced global illumination*. AK Peters Ltd, 2006. 2

[DLR*77] DEMPSTER A., LAIRD N., RUBIN D., ET AL.: Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. 39*, 1 (1977), 1–38. 2

[DYN04] DOBASHI Y., YAMAMOTO T., NISHITA T.: Radiosity for point-sampled geometry. In *12th Pacific Conference on Computer Graphics and Applications* (Oct. 2004), pp. 152–159. 2

[GNN10] GARCIA V., NIELSEN F., NOCK R.: Hierarchical Gaussian mixture model. In *Proceedings of the IEEE Interna-*

tional Conference on Acoustics, Speech, and Signal Processing (2010), pp. 4070–4073. 2, 6, 7

[GR05] GOLDBERGER J., ROWEIS S.: Hierarchical clustering of a mixture model. *Advances in Neural Information Processing Systems 17*, 505-512 (2005), 2–4. 2, 7

[HE03] HOPF M., ERTL T.: Hierarchical splatting of scattered data. In *Proceedings of VIS* (2003). 2

[HG09] HASAN B. A. S., GAN J.: Sequential EM for unsupervised adaptive Gaussian mixture model based classifier. In *Machine Learning and Data Mining in Pattern Recognition*, vol. 5632. 2009, pp. 96–106. 2

[HJJ10] HACHISUKA T., JAROSZ W., JENSEN H. W.: A progressive error estimation framework for photon density estimation. *ACM Transactions on Graphics. 29* (December 2010). 10

[HLE04] HOPF M., LUTTENBERGER M., ERTL T.: Hierarchical splatting of scattered 4d data. *IEEE Computer Graphics and Applications 24* (July 2004), 64–72. 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 130:1–130:8. 2, 10

[HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A rapid hierarchical radiosity algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 91)* (July 1991), pp. 197–206. 2

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Transactions on Graphics. 26*, 3 (2007). 2, 9

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of SIGGRAPH.* (July 1998). 1, 3

[JNSJ11] JAROSZ W., NOWROUZEZAHRAI D., SADEGHI I., JENSEN H. W.: A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics 30*, 1 (Jan. 2011), 5:1–5:19. 3, 9

[JZJ08] JAROSZ W., ZWICKER M., JENSEN H. W.: The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum 27*, 2 (Apr. 2008). 1, 3

[LH91] LAUR D., HANRAHAN P.: Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Proceedings of SIGGRAPH.* (July 1991), pp. 285–288. 2

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *Compugraphics* (1993). 1

[LW96] LAFORTUNE E. P., WILLEMS Y. D.: Rendering participating media with bidirectional path tracing. In *Eurographics Rendering Workshop* (June 1996), pp. 91–100. 1

[LZT*08] LEHTINEN J., ZWICKER M., TURQUIN E., KONTKANEN J., DURAND F., SILLION F. X., AILA T.: A meshless hierarchical representation for light transport. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 37:1–37:9. 2

[NH98] NEAL R., HINTON G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models 89* (1998), 355–368. 2, 4

[PJJ*11] PAPAS M., JAROSZ W., JAKOB W., RUSINKIEWICZ S., MATUSIK W., WEYRICH T.: Goal-based caustics. *Computer Graphics Forum (Proceedings of Eurographics '11) 30*, 2 (June 2011). 2

[PKK00] PAULY M., KOLLIG T., KELLER A.: Metropolis light transport for participating media. In *Eurographics Workshop on Rendering* (June 2000), pp. 11–22. 1

[RL00] RUSINKIEWICZ S., LEVOY M.: Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH.* (July 2000), pp. 343–352. 2

[SFES07] SCHJØTH L., FRISVAD J. R., ERLEBEN K., SPORRING J.: Photon differentials. In *GRAPHITE.* (2007), pp. 179–186. 2

[Sil86] SILVERMAN B. W.: *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1986. 2

[Sil95] SILLION F. X.: A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics 1*, 3 (1995). 2

[SJ09a] SPENCER B., JONES M. W.: Hierarchical photon mapping. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (Jan./Feb. 2009), 49–61. 2

[SJ09b] SPENCER B., JONES M. W.: Into the blue: Better caustics through photon relaxation. *Computer Graphics Forum 28*, 2 (Apr. 2009), 319–328. 2

[SSO08] SCHJØTH L., SPORRING J., OLSEN O. F.: Diffusion based photon mapping. *Computer Graphics Forum 27*, 8 (Dec. 2008), 2114–2127. 2

[TLQ*05] TAN P., LIN S., QUAN L., GUO B., SHUM H.-Y.: Multiresolution reflectance filtering. In *Proceedings Eurographics Symposium on Rendering 2005* (2005), pp. 111–116. 2, 9

[VG94] VEACH E., GUIBAS L.: Bidirectional estimators for light transport. In *Eurographics Workshop on Rendering* (1994). 1

[VNV06] VERBEEK J. J., NUNNINK J. R., VLASSIS N.: Accelerated EM-based clustering of large data sets. *Data Mining and Knowledge Discovery 13*, 3 (November 2006), 291–307. 2, 4

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics. 25*, 3 (2006). 1, 2

[WBKP08] WALTER B., BALA K., KULKARNI M., PINGALI K.: Fast agglomerative clustering for rendering. In *IEEE Symposium on Interactive Ray Tracing* (August 2008), pp. 81–86. 6

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics. 24*, 3 (Aug. 2005). 2

[ZHG*07] ZHOU K., HOU Q., GONG M., SNYDER J., GUO B., SHUM H.-Y.: Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. In *Pacific Graphics.* (2007), pp. 116–125. 2

[ZPvBG02] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: EWA splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (July/Sept. 2002), 223–238. 2

[ZRL*08] ZHOU K., REN Z., LIN S., BAO H., GUO B., SHUM H.-Y.: Real-time smoke rendering using compensated ray marching. *ACM Transactions on Graphics. 27*, 3 (2008). 2, 6