Unbiased, Adaptive Stochastic Sampling for Rendering Inhomogeneous Participating Media

Yonghao Yue[1] Kei Iwasaki[2] Bing-Yu Chen[3] Yoshinori Dobashi[4] Tomoyuki Nishita[1]
[1]The University of Tokyo [2]Wakayama University [3]National Taiwan University [4]Hokkaido University

Abstract

Realistic rendering of participating media is one of the major subjects in computer graphics. Monte Carlo techniques are widely used for realistic rendering because they provide unbiased solutions, which converge to exact solutions. Methods based on Monte Carlo techniques generate a number of light paths, each of which consists of a set of randomly selected scattering events. Finding a new scattering event requires free path sampling to determine the distance from the previous scattering event, and is usually a time-consuming process for inhomogeneous participating media. To address this problem, we propose an adaptive and unbiased sampling technique using kd-tree based space partitioning. A key contribution of our method is an automatic scheme that partitions the spatial domain into sub-spaces (partitions) based on a cost model that evaluates the expected sampling cost. The magnitude of performance gain obtained by our method becomes larger for more inhomogeneous media, and rises to two orders compared to traditional free path sampling techniques.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.3.3 [Computer Graphics]: Picture/Image Generation; G.3 [Probability and Statistics]: Probabilistic Algorithms

Keywords: Participating media, free path sampling, space partitioning, Monte Carlo technique, unbiased

1 Introduction

Realistic rendering of inhomogeneous participating media is very common, as there are many related phenomena in our daily life, *e.g.*, steam, water, fire, smoke, explosions, volcanic eruptions, clouds, atmosphere, mist due to waterfalls, splashes due to ocean waves, etc. These phenomena are the subjects in fluid simulation research and frequently appear in movies. Thus, their photo-realistic visualization is an important topic in both academic research and film industry.

The photo-realistic rendering of participating media is typically realized by techniques based on Monte Carlo sampling [Lafortune and Willems 1996; Jensen and Christensen 1998; Pauly et al. 2000; Raab et al. 2006]. In these methods, the intensity at a pixel is calculated by integrating contributions from a number of light paths, and the process to generate light paths is at the heart of these methods.

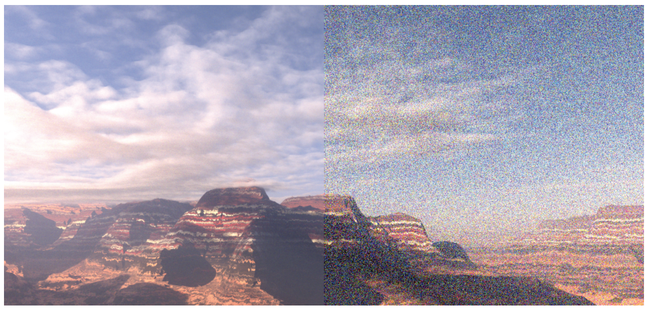


Figure 1: *Our method (left half) is able to generate 380 times more light paths than [Raab et al. 2006] (right half) in the same computation time, and the noise is significantly reduced. Sunlight is the only light source. Global illumination effects, including multiple scattering inside the clouds and the atmosphere, the haze, and shadows on the ground due to the clouds, are simulated.*

For inhomogeneous participating media, however, generating such light paths is usually a time-consuming process.

In this paper, we focus on an efficient method for generating the light paths. An important feature in designing such a method is the unbiasedness, which brings several benefits: 1) the solution provably converges to the exact one; 2) the computational error can be easily evaluated by measuring the variance, whereas the computational error in biased methods is difficult to evaluate [Veach 1998]; 3) there is no need to be concerned about bias-induced artifacts.

A light path in a participating medium is constructed by randomly generating successive scattering events. To determine the location of a new scattering event, the distance from the previous scattering event, called the *free path*, needs to be determined using random numbers. An importance sampling technique, called *free path sampling*, is often employed to efficiently determine the free path according to the probability density function which corresponds to the optical depth of the participating medium. The ray-marching approach is often employed [Pauly et al. 2000], but results in a biased solution. An unbiased solution can be obtained by using *Woodcock tracking* [Woodcock et al. 1965], which was proposed in the nuclear science community and is frequently used in nuclear science and medical physics [Badal and Badano 2009]. It was first introduced to the computer graphics community by [Raab et al. 2006]. The free path is sampled by incrementing small distances in a stochastic manner judging whether a scattering event has occurred. The lengths of such small distances are adjusted to be short enough to sample the densest region of the medium. However, Woodcock tracking becomes inefficient in inhomogeneous media [Leppänen 2007], because the mean free paths are much longer in sparse regions, and such small distances are usually incremented many times, ranging from tens to thousands, until the next scattering event occurs.

To overcome the above problem, we extend Woodcock tracking, and propose an adaptive and unbiased technique. During preprocessing, our method partitions the analytical space (the bounding box of the medium) into sub-spaces (partitions) according to the spatial variation of the mean free path in the medium. The parti-

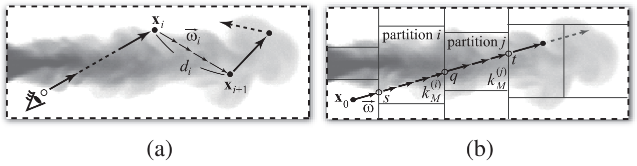


Figure 2: *(a): Illustration of light paths. (b): Sampling in partitioned space. Black dots show the locations of scattering events and the boxes with dashed lines show the bounding box of the medium.*

Algorithm 1: WoodcockTracking($\mathbf{x}_0$, $\vec{\omega}$, k_M, d_{min}, d_{max})

```
Input: $\mathbf{x}_0, \vec{\omega}$ : The ray starting at $\mathbf{x}_0$ in direction $\vec{\omega}$.
    $k_M$ : The majorant extinction coefficient.
    $(d_{min}, d_{max}]$ : The interval of the ray to evaluate.
Output: The free path $d$ to the next scattering event.
1: $d \leftarrow d_{min} - \ln(1 - rand())/k_M$
2: while $d \le d_{max} \wedge k(\mathbf{x}_0 + d\vec{\omega})/k_M < rand()$ do
3:     $d \leftarrow d - \ln(1 - rand())/k_M$
4: end while
5: return $d$
```

tioning is represented as a kd-tree. During rendering, the locations of the scattering events are determined adaptively using the kd-tree. Our sampling technique is proven to be unbiased. A key contribution of our method is an automatic partitioning scheme based on a cost model for evaluating the sampling efficiency. We find the optimal partitioning with respect to the cost model by solving the *largest empty rectangle problem*. Another contribution of our method is its scalability to handle a wide range of scenes including clouds and atmosphere in whole-sky-scale as shown in Figure 1. Using our method, the cost for obtaining the free path becomes as low as the cost for other rendering processes including shading, contrasting with other methods where it is the dominant cost. Our method is one to two orders of magnitude faster in the overall rendering speed than the previous methods for highly inhomogeneous media, and is implemented on the GPU using CUDA.

An important benefit of our method is the ability to accelerate all of the previous methods that generate light paths for rendering participating media. Such methods include path tracing and Metropolis light transport [Pauly et al. 2000], as well as photon mapping [Jensen and Christensen 1998]. This benefit is also applicable in other fields, such as nuclear science and medical physics.

2 Related Work

There has been much previous research on rendering participating media. Thorough reviews can be found in [Cerezo et al. 2005] or [Gutierrez et al. 2009]. Here, we briefly review some closely related papers. According to Cerezo et al. [2005], rendering methods for participating media can be classified into voxel-based methods [Stam 1995; Nishita et al. 1996] and sampling-based methods. Voxel-based methods are usually inefficient for obtaining accurate results, as the resolution of the voxels needs to be maintained finely.

In sampling-based methods, such as Metropolis light transport [Pauly et al. 2000] and photon mapping [Jensen and Christensen 1998], the intensity at a pixel is calculated by generating a number of light paths, each of which consists of a set of scattering events. Major techniques to determine the locations of the scattering events are the ray-marching method [Perlin and Hoffert 1989; Jensen and Christensen 1998; Pauly et al. 2000] or its variants, *e.g.*, [Brown and Martin 2003]. However, they are biased and will produce different results for different sampling intervals.

For unbiased sampling, Raab et al. [2006] introduced Woodcock tracking [Woodcock et al. 1965; Lux and Koblinger 1991] (also known as delta-tracking or pseudo-scattering) from the field of nuclear science. Although it has been proven to be unbiased [Coleman 1968], it is known to be inefficient when dealing with inhomogeneous media [Leppänen 2007]. Surprisingly, however, little work has been proposed to overcome this inefficiency in the past few decades. Recently, GPU implementations of Woodcock tracking were proposed by, *e.g.*, Badal et al. [2009]. Leppänen [2007] improved Woodcock tracking by separating dense regions from sparse regions, and treating these two kinds of regions differently. Unlike the problem settings in Leppänen [2007], the density distribution of participating media appearing in the computer graphics field (such as smoke and clouds) is usually continuous, and it is not obvious how to adapt such two-level separation.

In this paper, we propose for the first time an automatic space partitioning scheme and use a kd-tree to represent the partitioning. Inefficiency for rendering inhomogeneous participating media is overcome by utilizing the space partitioning for free path sampling. Moreover, our method scales well to the complexity of the scene.

3 Free Path Sampling

We first state the free path sampling problem, and then discuss the issues of ray-marching and Woodcock tracking.

Monte Carlo techniques calculate the pixel intensity by first generating a number of light paths, and then integrating their contributions. To generate a light path, scattering events are generated successively as shown in Figure 2(a). Assume the scattering event i has already been generated. The location of scattering event $(i+1)$ is typically determined by sampling the free path d_i and the scattering direction $\vec{\omega}_i$ by using random numbers. For the scattering direction, we can use conventional importance sampling. The free path d_i should be sampled according to the following probability density function [Pauly et al. 2000]:

$$pdf_{fp}(\mathbf{x}_{i+1} = \mathbf{x}_i + d_i\vec{\omega}_i) = e^{-\tau(\mathbf{x}_i,\mathbf{x}_{i+1})}k(\mathbf{x}_{i+1}), \qquad (1)$$

where $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ are the locations of the scattering events i and $(i+1)$. $\tau(\mathbf{x}_i, \mathbf{x}_{i+1}) = \int_{\mathbf{x}_i}^{\mathbf{x}_{i+1}} k(\mathbf{x}')d\mathbf{x}'$ is the optical depth between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. k indicates the extinction coefficients. To generate random numbers that obey the above probability density function, the inversion method is usually used together with ray-marching: first, a random number $\xi \in (0, 1)$ is drawn using $rand()$, then d_i satisfying $\tau(\mathbf{x}_i + d_i\vec{\omega}_i, \mathbf{x}_i) = -\ln(1-\xi)$ is found using ray-marching, which, however, introduces bias.

Woodcock tracking (shown as Algorithm 1) samples the free path in an unbiased way by utilizing a rejection sampling technique. First, a *majorant* extinction coefficient k_M which is the maximum extinction coefficient of the participating medium is computed. Then, Woodcock tracking samples pseudo scattering events by regarding the medium as a uniform medium with the extinction coefficient being k_M. For unbiasedness, such pseudo scattering events are only accepted as 'real' scattering events with the probability $k(\mathbf{x}_i + d_i\vec{\omega}_i)/k_M$. This process is proven to be unbiased and we can obtain d_i with the probability density function given by Equation (1) [Coleman 1968]. Every time a random number is generated, the free path d_i (see Figure 2(a)) is incremented by $-\ln(1 - rand())/k_M$, whose expectation value is $1/k_M$. In an inhomogeneous participating medium, $k(\mathbf{x}_i + d_i\vec{\omega}_i)$ is often much smaller than the majorant extinction coefficient k_M, and the ratio $k(\mathbf{x}_i + d_i\vec{\omega}_i)/k_M$ becomes small where the medium is sparse. Therefore, many random numbers might be generated until a real scattering event is detected. Thus, this method becomes less efficient when the participating medium is highly inhomogeneous.

4 Unbiased Sampling and Space Partitioning

To efficiently sample the free path, we partition the bounding box containing the participating medium into some partitions. Then the majorant extinction coefficient for each partition can be set smaller than that for the entire domain. The results of this are that 1) the average distance in each iteration ($1/k_M$) becomes longer, and 2) scattering events are more likely to be accepted. Thus, the expected total number of iterations can be reduced.

A simple way to do this partitioning is to use a uniform grid [Szirmay-Kalos et al. 2010]. However, the resolution of the grid significantly influences the sampling performance and needs to be tuned manually. Moreover, uniform grids do not scale well to the size of the scene. Instead, we present an automatic partitioning scheme and use a kd-tree to represent the partitioning (Figure 2(b)). The partitioning adapts well to the size of the scene and the sampling performs much better than using a uniform grid.

Firstly, we discuss how to sample the free path in an unbiased manner when the partitioning is given. Then, we discuss our space partitioning strategy that makes our sampling technique efficient. Finally, we summarize our sampling technique utilizing the kd-tree.

4.1 Unbiased Sampling in Partitioned Space

Suppose that a ray starting at $\mathbf{x}_0$ in the direction $\vec{\omega}$ (Figure 2(b)) passes through two adjacent partitions i and j, with $k_M^{(i)}$ and $k_M^{(j)}$ being their majorant extinction coefficients. Let the distances from $\mathbf{x}_0$ to the intersections with those partitions be s, q, and t.

To sample the free path in the interval $(s, t]$, we first deal with the first interval $(s, q]$ by using Algorithm 1 with k_M, d_{min}, and d_{max} set to $k_M^{(i)}$, s and q. We may obtain a scattering event in the first interval or the free path exceeds q otherwise. If the free path exceeds q, we rewind the free path back to q and proceed to the second interval $(q, t]$, and Algorithm 1 is used with $k_M^{(j)}$, q, and t. By rewinding the free path back to q, we can ensure the unbiasedness of the sampling. This strategy of rewinding was also used in [Carter et al. 1972] to treat adjacent medium boundary. We provide a simple proof of the unbiasedness in Appendix A, by showing that we can obtain the free path d with the probability:

$$\begin{cases} P(\mathbf{x}' = \mathbf{x}_0 + d\vec{\omega} \wedge s < d \le t) = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ P(d > t) = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}_0 + t\vec{\omega})} \end{cases}, \quad (2)$$

which is equivalent to using Algorithm 1 with d_{min} and d_{max} being s and t, respectively, and $k_M = \max_{s \le z \le t} k(\mathbf{x}_0 + z\vec{\omega})$.

4.2 Partitioning Participating Media

The performance of the sampling technique shown in Section 4.1 highly depends on the partitioning. If we partition a nearly homogeneous interval, an additional iteration is required to rewind the free path to q when proceeding to the adjacent interval, and this would be a waste compared to the case with no partitioning. On the other hand, we can have more benefit, if $k_M^{(i)}$ in a partition is much smaller than k_M for the entire space. These observations would give us a stopping criterion for the partitioning process, as well as a decision criterion for the partitioning location q. We first explain our method for a one dimensional case, and discuss the optimal partitioning location. Then, we extend the method to three-dimensions.

One-dimensional case. Suppose that $k(x)$ is given at an arbitrary location x. Let the w axis indicate the value of $k(x)$, and we consider an interval $(s, t]$. First, we consider a simple case as shown

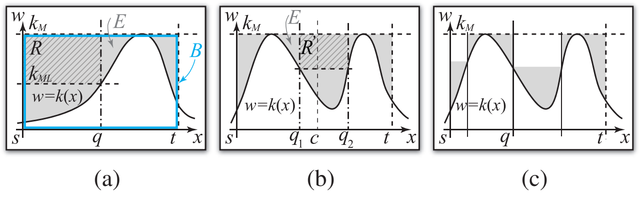


Figure 3: *Example distributions of media in 1D.*

in Figure 3(a), where $k(x)$ has only one local maximum in $(s, t]$. We decide whether we should partition this interval as well as the optimal partitioning location through a cost model evaluating the expected number of iterations as follows. Let k_M be the majorant extinction coefficient in this interval. Then, in a single iteration, the free path will proceed $1/k_M$ on average. Therefore, we need $N = (t - s)k_M$ iterations, on average, to go across this interval. If we partition this interval at location q, as in Figure 3(a), it takes $N_{part} = (q - s)k_{ML} + (t - q)k_M + 1$ iterations, on average, to go across the entire interval $(s, t]$, where k_{ML} is the majorant extinction coefficient in the interval $(s, q]$. The last term '$+1$' arises because we need to reset the location to q when we proceed to the second interval. Then, the reduction N_r in the expected number of iterations when we partition the interval $(s, t]$ at q is given by

$$N_r(q) = N - N_{part} = (q - s)(k_M - k_{ML}) - 1. \quad (3)$$

If $N_r \le 0$, we cannot gain any benefit, and the interval is left unpartitioned. Otherwise the optimal partitioning location can be obtained by finding q that maximizes the reduction, *i.e.*,

$$q = \operatorname*{argmax}_{q' \in (s,t]} N_r(q'). \quad (4)$$

Next, we generalize the above discussion to handle the cases where $k(x)$ has multiple peaks like Figure 3(b) by replacing Equation (4) with a problem to find the largest rectangle R in Figure 3(a). By inspecting Figure 3(a), we notice that $(q-s)(k_M - k_{ML})$ on the right hand side of Equation (3) corresponds to the area of a rectangle R (hatched region in Figure 3(a)), and R should touch the boundary of the empty space E (the light gray region bounded by k_M, $k(x)$, $x = s$ and $x = t$). Similarly, the expected number of iterations to traverse the interval $(s, t]$ without the partitioning, $(t - s)k_M$, is exactly the area of the bounding box B (the blue rectangle bounded by $w = k_M$, $w = 0$, $x = s$ and $x = t$). Intuitively, the bounding box B contains empty space E, which corresponds to the waste in sampling. By finding the largest rectangle R and partitioning the interval at q, we can remove the waste corresponding to R.

To define the problem formally, we replace Equation (4) with

$$R = \operatorname*{argmax}_{R' \subset E} N^{\Delta}(R'), \quad (5)$$

where N^{Δ} is the reduction in the expected number of iterations corresponding to removing the rectangle R' from E, and is given by $N^{\Delta}(R') = A(R') - T(R')$. $A(R')$ is the reduction in the expected number of iterations when we simply remove R' from E, and is exactly the area of R'. $T(R')$ is the number of additional iterations we have to pay when the interval is partitioned, and is determined according to the alignment of R'. $T(R') = 2$ if both lower corners of R' are on $w = k(x)$ (Figure 3 (b)), since we have to partition the interval at both ends of R' to remove R' from E. Otherwise, $T(R') = 1$ (Figure 3 (a)). We call $N^{\Delta}(R')$ the *reduced area* of R'.

To find R that satisfies Equation (5), we solve a slightly modified version of the 2D largest empty rectangle problem [Aggarwal and

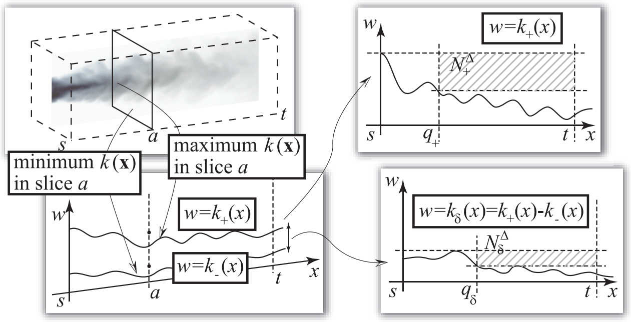


Figure 4: *An illustration of building functions $k_+(x)$ and $k_\delta(x)$, and calculating partitioning coordinates. For each location a, we scan along a slice 'a' perpendicular to the x axis, and find the maximum and minimum extinction coefficients ($k_+(a)$ and $k_-(a)$, respectively) in this slice. $k_\delta(a)$ is then defined as $k_+(a) - k_-(a)$.*

Suri 1987]. In our problem, the area of each rectangle is replaced by the reduced area. For the solution, we discretize the empty region E into n bins along the x axis and use dynamic programming to find the largest empty rectangle in E. Both the computational complexity and the memory requirement are in $O(n)$. For details, please refer to the supplemental material.

Assume R' in Figure 3(b) is reported to be the largest rectangle with the reduced area. Let q_1 and q_2 be the two ends of the rectangle. As we use a kd-tree to represent the partitions, we partition the interval once. Of these two ends, the one that is closer to the center $c = (s+t)/2$ of interval $(s, t]$ is chosen as the partitioning location q as shown in Figure 3(c). This makes the lengths of the two new intervals, $(s, q]$ and $(q, t]$ as nearly equal as possible. The partitioning is then recursively applied to each of the two child intervals, as shown in Figure 3(c). The kd-tree is constructed so that an inner node represents the corresponding partitioning location and a leaf node stores the majorant extinction coefficient of the corresponding interval.

Three-dimensional case. An interval of the medium now becomes a volumetric region V. Hence, we want to choose a plane for partitioning V. To maintain a kd-tree, we only consider planes aligned to axes for partitioning. A straightforward way to find the partitioning plane is to find the largest hyper-rectangle in the 4D region bounded by $w = k(\mathbf{x}), w = k_M$ and V. However, current solutions for the largest empty rectangle problem in 4D space have high computational complexity [Edmonds et al. 2003].

Instead, we employ a heuristic approach, with which, in our experiments, the kd-tree can be built in less than one minute. Although it is not always optimal, it works well for media with complex distributions as shown in Section 5.2. We regard the problem as three individual 1D problems along each of the three axes, and try to find the partitioning candidate for each of the axes. We then choose the best one from those candidates comparing their benefits.

We first explain how to find the partitioning candidate for the x axis. We first setup a 1D function $k_+(x)$ along the x axis, defined as $k_+(x) = \max_{y,z,(x,y,z)\in V} k(x, y, z)$. We apply the solution of the one dimensional case to find the largest empty rectangle and obtain its reduced area N_+^Δ. If $N_+^\Delta > 0$, we obtain the corresponding partitioning coordinate q_+. This simple approach works well for most cases, but if $k(\mathbf{x})$ has many peaks in V and their projections onto 1D overlap each other, $k_+(x)$ tends to become uniform and fails to detect the non-uniform regions, *e.g.*, sparse regions that are enclosed by dense regions. To address this situation, we introduce another 1D function $k_\delta(x)$, defined as $k_\delta(x) = \max_{y,z,(x,y,z)\in V} k(x, y, z) - \min_{y,z,(x,y,z)\in V} k(x, y, z)$. Please see Figure 4 for a graphical illustration of $k_+(x)$ and $k_\delta(x)$. Using $k_\delta(x)$ makes it possible to detect the non-uniform regions in such a case, because $k_\delta(x)$ becomes large when the distribution of the medium in the plane at x perpendicular to the x axis is inhomogeneous and becomes small if the distribution of the medium is nearly homogeneous. We notice that the 1D largest empty rectangle solver can be used to detect such homogeneous regions. Similar to the process for obtaining N_+^Δ from $k_+(x)$, we obtain N_δ^Δ from $k_\delta(x)$. If $N_\delta^\Delta > 0$, we obtain another partitioning coordinate q_δ. The last step is then to choose the partitioning candidate from these two coordinates. Empirically, we find that using q_+ preferentially will result in better performance. Thus, we compare N_+^Δ with $F \cdot N_\delta^\Delta$, where $F < 1$ is a constant. We then choose the larger one and let its value be the numerical representation of the benefit for the x axis. We find that $F = 0.7$ gives good performance.

The above process is performed for the y and z axes. To lower the cost for free path sampling, we seek the largest benefit. If it is less than or equal to 0, V is left unpartitioned. Otherwise, we select the partitioning axis and location that give the largest benefit value.

The memory footprint of a kd-tree is usually negligible compared to the storage for the participating medium itself. In our implementation, the memory layout of the kd-tree is similar to that for standard ray tracing [Wald 2004], and each node consumes only 8 bytes.

To handle object surfaces, we first perform the above process ignoring the surfaces. Then, for each leaf node, if surfaces are aligned in the node, we further construct a sub kd-tree in the node for the surfaces. Compared to using separate kd-trees for the surfaces and the medium, we can reduce redundant ray-surface intersections.

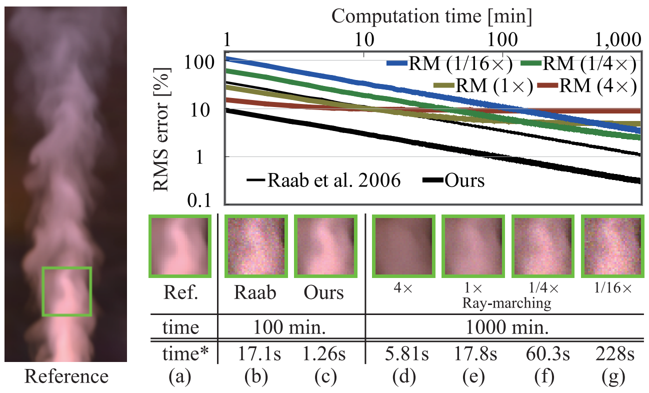
Algorithm 2: kdTreeFreePathSampling($\mathbf{x}_0$, $\vec{\omega}$)

Input: $\mathbf{x}_0$, $\vec{\omega}$: The ray starting at $\mathbf{x}_0$ in direction $\vec{\omega}$.
Output: The free path d to the next scattering event.

```
 1: loop
 2:     kdTreeNode $p$ ← findNextLeafNode()
 3:     if $p$ = nil then  return INFINITY  end if
 4:     $(d_{min}, d_{max})$ ← intersectionBetweenRayAndNode($p$)
 5:     $k_M$ ← majorantExtinctionCoefficientStoredIn($p$)
 6:     $d$ ← WoodcockTracking($\mathbf{x}_0$, $\vec{\omega}$, $k_M$, $d_{min}$, $d_{max}$)
 7:     $d_{isect}$ ← INFINITY
 8:     if intersectionWithObjectSurfaces($\mathbf{x}_0$, $\vec{\omega}$, $p$) then
 9:         $d_{isect}$ ← distanceToTheIntersection()
10:     end if
11:     if $d \geq d_{isect}$ then  return IntersectionEvent  end if
12:     if $d < d_{max}$ then  return $d$  end if
13: end loop
```

4.3 Ray Traversal

Algorithm 2 shows a ray traversal algorithm using the kd-tree. The algorithm proceeds by first locating the leaf node for the current sampling location. d_{min} and d_{max} in line 4 indicate the distances to the nearest and farthest points of the node. In each leaf node, we apply Algorithm 1 to sample the location of the scattering event. If a ray-surface intersection is found as shown in line 8, d_{isect} is set to the distance to the intersection. If d obtained in line 6 exceeds d_{isect}, we return an intersection event. If the location corresponding to d is inside the current leaf node, that location is determined as the scattering location. Otherwise, we continue to the next interval, by traversing the next leaf node. The resulting algorithm is quite simple and similar to the standard kd-tree based ray tracing algorithm for polygonal scenes, *e.g.*, [Keller 1998]. Lines 2, 4, 8 and 9 of Algorithm 2 share his code.



Ref.	Raab	Ours	4×	1×	1/4×	1/16×
time	100 min.		1000 min.			
time*	17.1s	1.26s	5.81s	17.8s	60.3s	228s
(a)	(b)	(c)	(d)	(e)	(f)	(g)

Figure 5: *(a): Close view of the reference image. (b) to (c): rendered using [Raab et al. 2006] and our method with 100 minutes. (d) to (g): rendered with 1,000 minutes using ray-marching with the sampling interval being $4\times$, $1\times$, $1/4\times$ and $1/16\times$ of $1/k_M$ of the medium, respectively. time*: The time to generate 10^6 rays.*

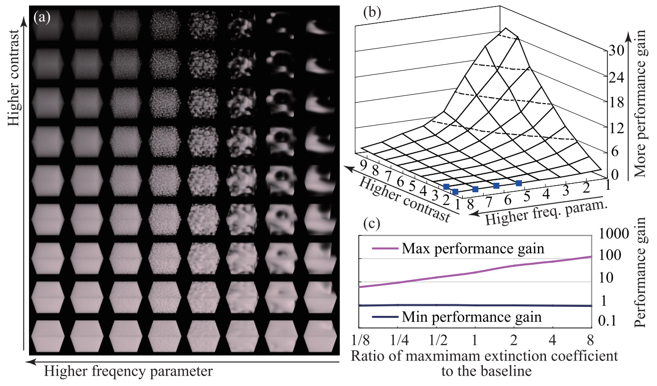


Figure 6: *(a): The appearance of the set of media with maximum extinction coefficient being equal to the baseline. (b): The performance gains for the set. Blue squares in (b) indicate inferior cases. (c): The max/min performance gains for each of the set.*

5 Evaluations

We conducted three types of experiments: 1) comparison of rates of convergence, 2) detailed investigation of the performance gain and 3) rendering natural participating media. All the tests were performed on a single PC with an Intel Core2 Extreme QX9650 CPU and an nVIDIA GeForce GTX 295 GPU. Tests in Sections 5.1 and 5.2 were performed on a single core of the CPU without SIMD instructions for fairness. To simplify the evaluation, we implemented our sampling technique in a simple Monte Carlo path tracing method. Multiple scattering and inter-reflection of light were computed in all of the tests, and Russian roulette was used to determine the termination of the light paths. In all of the experiments, the computation time for the space partitioning was less than one minute, and this is negligible compared to the time needed for obtaining a result with the noise reasonably reduced.

5.1 Discussion on Convergence

We compared the rates of convergence between [Raab et al. 2006], our method, and ray-marching in Figure 5. We applied random offsets to the first sampling interval for ray-marching to avoid aliasing artifacts as in [Pauly et al. 2000]. The left image is the reference image rendered using [Raab et al. 2006] in 5 days with 16 cores. The top right graph shows the rates of convergence by plotting the RMS error against the computation time on a log-log scale.

The results rendered using ray-marching do converge, but to wrong solutions. This can be seen by comparing (a) with (d), and from the fact that the graphs of ray-marching are 'curving out' which is clearly noticeable when the sampling interval is 4x and 1x of $1/k_M$. The bias can be reduced by shortening the sampling interval, but more time is needed for a converged result. By contrast, the result rendered using our method converges to the correct one, and the performance is superior to all of the other cases. Comparing the computation times required to generate 10^6 rays, our method is 13.6 times faster than [Raab et al. 2006]. This gain results in faster reduction of the noise due to the Monte Carlo estimation. The noise is clearly visible in (b), but has almost disappeared in (c).

5.2 Detailed Evaluation of Performance Gain

We compared the computation times of free path sampling for various media using Algorithms 1 and 2. To cover a variety of media, we changed the degree of concentration (extinction coefficient), size and inhomogeneity. Since a large scene with low optical coefficients (extinction, scattering and absorption coefficients) and a small scene with high optical coefficients will have the same appearance, we fix the scale of length and only consider the scale of optical coefficients. Inhomogeneity is controlled by changing the frequency parameter of 3D Perlin noise [Perlin 2002] (higher frequency parameter means more rapid local variation in the medium).

We first prepared 8 types of base media by setting the frequency parameter from 1 to 8. We then normalized the extinction coefficients in the range $[0, 1]$. Then, we prepared 9 types of variations for each base medium by changing the contrast to $2^j, (j = 1, ..., 9)$, where the contrast is the ratio of the maximum difference of the extinction coefficient to the average extinction coefficient. To obtain a desired contrast, we applied the power r to the normalized extinction coefficients, where r is a real number and is obtained using a bisection method. Finally, for each of these 72 types of media, we changed the maximum extinction coefficient to $1/8, 1/4, 1/2, 1, 2, 4, 8$ times of the baseline to create 7 sets of sparser and denser media. Then, we examined the performance for these 504 types of media.

As Algorithms 1 and 2 sample the free path with the same probability distribution, their computation time for generating 10^6 rays can be compared fairly. Figure 6 (b) shows the performance gains of Algorithm 2 over Algorithm 1 for the set of media with the maximum extinction coefficient being equal to the baseline, and their appearances are shown in Figure 6 (a). The tendency of the performance gain for the other 6 sets of media is similar. Therefore, instead of displaying all of the performance gains, we show the relationship between the maximum extinction coefficient, and the maximum / minimum performance gains in Figure 6 (c). We notice that more performance gains are obtained for media with higher maximum extinction coefficients.

The performance gains are higher when the media have higher contrasts and are generated from base media with lower frequency parameters. If high concentration regions are congregated in a narrower range as shown in Figure 6 (a), our method is more efficient. Natural participating media, such as smoke, clouds and fire often have this property. Although there are a few inferior cases shown as the blue squares in Figure 6 (b), our method is faster than [Raab et al. 2006] in most cases. The inferior cases were all reported for nearly homogeneous media, and the performance loss is at most a few percent. Such performance loss happens because of the fol-

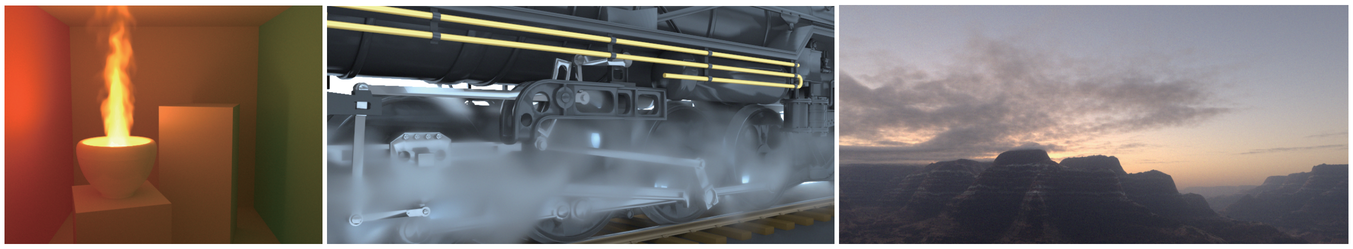


Figure 7: *Rendered results of inhomogeneous participating media using our method. Global illumination effects, including multiple scattering and inter-reflection, are simulated. Image sizes are 640 by 480 for the left image and 960 by 450 for the middle and right images.*

lowing two reasons: 1) the heuristics are imperfect; 2) the cost of traversing the kd-tree influences the performance slightly.

Next, we would like to briefly mention the performance gain for the whole rendering process, which have other sub-processes, such as shading and sampling the scattered direction. In our implementation, they take up about half of the computation time. Thus, the performance gain for the whole rendering process is about half of that for free path sampling.

5.3 Rendered Results

The images in Figure 7 from left to right show the fire, steam and sky scenes, lit by the fire, an environment light source and the sunlight, respectively. Scene statistics are shown in Table 1. Kd-tree construction took 5, 7 and 42 seconds for the fire, steam and sky scenes. Our method is 2.7, 8.9 and 380 times faster than [Raab et al. 2006] to render the images in the same quality for these scenes. These increased speeds come from the fact that the numbers of average iterations are significantly reduced. Our method running on the GPU is about 50 times faster than on a single core of the CPU. At each thread on the GPU, we compute tens of light paths for a single pixel: light paths passing through the pixel are started from the viewpoint, and successive scattering or intersection events are generated. A single CUDA kernel execution evaluates the contributions for all the pixels. We execute the kernel hundreds of times, and then finally average the contribution of each light path to obtain the result. The rendering times for these scenes are 225 min, 58 min and 345 min on the GPU. Using a two-layered kd-tree described in Section 4.2 resulted in 10 to 35% faster speedup than using separate kd-trees for the object surfaces and the participating medium.

In the sky scene, the clouds are in a voxel-based representation and is modeled based on a satellite image. The atmosphere is in a procedural representation. We used physically-based optical coefficients and phase functions to render the images. In the real world, the extinction coefficients of the clouds are about 10^2 to 10^3 times as large as those of the atmosphere, thus the sky is highly inhomogeneous. In Figures 1 and 7 right, global illumination effects, including multiple scattering inside the clouds and the atmosphere, the haze, and shadows on the ground due to the clouds, are simulated.

5.4 Discussion and Applications

On voxelization. If the participating medium is not in a voxelized representation, we need a temporal voxelized representation, *e.g.*, a uniform grid or an adaptive grid, during the space partitioning in order to solve the largest empty rectangle problem. In each voxel, the majorant extinction coefficient of the corresponding subspace needs to be stored. As long as the majorant extinction coefficient is any valid upper bound of the extinction coefficient (*i.e.*, not necessarily the least upper bound), the unbiasedness of our method is guaranteed. A tighter bound leads to more efficient sampling. If the participating medium is in a procedural representation, in practice, we can estimate an upper bound from the formulae of the procedural representation. Currently, the resolution is chosen by the user. We recommend the use of as high resolutions as possible for better performance in sampling.

Table 1: *Scene statistics. Values in the style 'x/y' show '#ray/sec' and '#average iterations'. $\overline{depth}$ and #leaf show the average depth and the number of leaves of the kd-tree for the participating media. $\overline{k}$ and $\sigma(k)$ show the average and standard deviation of the extinction coefficient.*

Scene	Performance (GPU)		Speed-up	$\overline{depth}$	*#leaf*	$\overline{k}$	$\sigma(k)$
	Raab et al. 2006	Our					
Fire	4.6M / 7.3	12M / 2.6	2.7	4.86	7	0.076	0.55
Steam	920K / 196	8.2M / 6.4	8.9	16.4	350	0.78	5.61
Sky	8.2K / 7070	3.12M / 8.8	380	13.9	193	*atmosphere*	
						3.2×10^{-6}	1.14×10^{-5}
						clouds	
						1.52×10^{-2}	1.82×10^{-2}

On the heuristics. As a quantitative evaluation of our heuristic approach, we prepared various volume data with two different resolutions, 8^3 and 16^3, and compared the computation time for the free path sampling using the kd-trees built with our heuristic approach and with the optimal partitioning solution. Comparisons for higher resolutions were not possible due to the prohibitive computation cost for the optimal solution. We found that use of the optimal solution is more efficient in sampling than using the heuristic approach, but the gain is only less than 10% on average.

Applications. Our unbiased free path sampling technique is a common building block for typical rendering methods based on Monte Carlo sampling. We would like to emphasize that our method can be used with any algorithm, including photon mapping, that must sample inhomogeneous media. The code for generating scattering events can be replaced by our method for better accuracy and performance. Our method is also applicable to problems in other fields: nuclear simulation in the nuclear science field, x-ray simulation in the medical physics field, etc. Additionally, our method can be used to accelerate unbiased computation of the transmittance, which also uses Woodcock tracking [Raab et al. 2006].

6 Conclusions and Future Work

We have proposed a free path sampling technique to efficiently find scattering events. Our method partitions the spatial domain into partitions that adapt to the inhomogeneous distribution of the medium, by solving the largest empty rectangle problem and using the kd-tree to represent the partitioning. Our sampling technique then samples the free path using the kd-tree in an unbiased manner. We have conducted a case study to investigate the performance gain of our method over previous methods, reporting one to two orders of magnitude acceleration for highly inhomogeneous media. By using our method, we can accelerate the rendering of many types of participating media, including the atmosphere and clouds, taking multiple scattering into account.

For future work, we would like to develop efficient solutions for the largest empty rectangle problem in high dimensional spaces. A 5D solution would also be important if we want to obtain a motion blur effect. Another possible extension is to incorporate other integrators, such as Metropolis light transport.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, Mikio Shinya and Masanori Kakimoto for thoughtful discussions, Yosuke Bando, Yoshihiro Kanamori, Paulo Silva and Pablo Garcia Trigo for proofreading. This work was supported in part by Grant-in-Aid for JSPS Fellows (20·7968).

References

AGGARWAL, A., AND SURI, S. 1987. Fast algorithms for computing the largest empty rectangle. In *Proc. Third Annual Symposium on Computational Geometry*, 278–290.

BADAL, A., AND BADANO, A. 2009. Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit. *Medical Physics 36*, 11, 4878–4880.

BROWN, F. B., AND MARTIN, W. R. 2003. Direct sampling of Monte Carlo flight paths in media with continuously varying cross-sections. In *Proc. ANS Mathematics & Computation Topical Meeting*.

CARTER, L., CASHWELL, E., AND W.M.TAYLOR. 1972. Monte Carlo sampling with continuously varying cross sections along flight paths. *Nuclear Science and Engineering 48*, 403–411.

CEREZO, E., PÉREZ, F., PUEYO, X., SERÓN, F. J., AND SILLION, F. X. 2005. A survey on participating media rendering techniques. *The Visual Computer 21*, 5, 303–328.

COLEMAN, W. 1968. Mathematical verification of a certain Monte Carlo sampling technique and applications of the technique to radiation transport problems. *Nuclear Science and Engineering 32*, 76–81.

EDMONDS, J., GRYZ, J., LIANG, D., AND MILLER, R. J. 2003. Mining for empty spaces in large data sets. *Theoretical Computer Science 296*, 3, 435–452.

GUTIERREZ, D., JENSEN, H. W., JAROSZ, W., AND DONNER, C. 2009. Scattering. In *SIGGRAPH ASIA Courses*, ACM.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proc. SIGGRAPH 98*, ACM, 311–320.

KELLER, A. 1998. *Quasi Monte Carlo methods for realistic image synthesis*. PhD thesis, University of Kaiserslautern.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1996. Rendering participating media with bidirectional path tracing. In *Proc. EGWR 96*, 91–100.

LEPPÄNEN, J. 2007. *Development of a New Monte Carlo Reactor Physics Code*. PhD thesis, Helsinki University of Technology.

LUX, I., AND KOBLINGER, L. 1991. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC Press.

NISHITA, T., DOBASHI, Y., AND NAKAMAE, E. 1996. Display of clouds taking into account multiple anisotropic scattering and sky light. In *Proc. SIGGRAPH 96*, ACM, 379–386.

PAULY, M., KOLLIG, T., AND KELLER, A. 2000. Metropolis light transport for participating media. In *Proc. EGWR 2000*, 11–22.

PERLIN, K., AND HOFFERT, E. M. 1989. Hypertexture. In *Computer Graphics (Proc. SIGGRAPH 89)*, ACM, vol. 23, 253–262.

PERLIN, K. 2002. Improving noise. In *Proc. SIGGRAPH 2002*, ACM, 681–682.

RAAB, M., SEIBERT, D., AND KELLER, A. 2006. Unbiased global illumination with participating media. In *Proc. Monte Carlo and Quasi-Monte Carlo Methods 2006*, 591–605.

STAM, J. 1995. Multiple scattering as a diffusion process. In *Proc. EGWR 95*, 41–50.

SZIRMAY-KALOS, L., TÓTH, B., MAGDICS, M., AND CSÉBFALVI, B. 2010. Efficient free path sampling in inhomogeneous media. In *Poster Proc. Eurographics 2010*.

VEACH, E. 1998. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University.

WALD, I. 2004. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University.

WOODCOCK, E., MURPHY, T., HEMMINGS, P., AND LONGWORTH, T. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conference on the Application of Computing Methods to Reactor Problems, ANL-7050*, 557–579.

A Proof of Unbiasedness of Our Method

It is proven by [Coleman 1968] that Woodcock tracking (Algorithm 1) can generate the free path d that obeys the probability distribution shown in Equation (2). We show below that our sampling technique can generate d with the same probability distribution. From the probability distribution of Algorithm 1, d_1 is obtained during the sampling for the first interval $(s, q]$ with the probability,

$$\begin{cases} P_1(\mathbf{x}' = \mathbf{x}_0 + d_1\vec{\omega} \wedge s < d_1 \le q) = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ P_1(d_1 > q) = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}_0 + q\vec{\omega})} \end{cases} . \tag{6}$$

Similarly, d_2 is obtained during the sampling for the second interval $(q, t]$ with the probability,

$$\begin{cases} P_2(\mathbf{x}' = \mathbf{x}_0 + d_2\vec{\omega} \wedge q < d_2 \le t) = e^{-\tau(\mathbf{x}_0 + q\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ P_2(d_2 > t) = e^{-\tau(\mathbf{x}_0 + q\vec{\omega}, \mathbf{x}_0 + t\vec{\omega})} \end{cases} . \tag{7}$$

Since the sampling for the second interval is executed if and only if d_1 exceeds q during the sampling for the first interval, we obtain

$$\begin{cases} P(\mathbf{x}' = \mathbf{x}_0 + d\vec{\omega} \wedge s < d \le q) \\ \quad = P_1(\mathbf{x}' = \mathbf{x}_0 + d\vec{\omega} \wedge s < d \le q) \\ \quad = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ P(\mathbf{x}' = \mathbf{x}_0 + d\vec{\omega} \wedge q < d \le t) \\ \quad = P_1(d > q) P_2(\mathbf{x}' = \mathbf{x}_0 + d\vec{\omega} \wedge q < d \le t) \\ \quad = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}_0 + q\vec{\omega})} e^{-\tau(\mathbf{x}_0 + q\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ \quad = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}')} k(\mathbf{x}') \\ P(d > t) = P_1(d > q) P_2(d > t) \\ \quad = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}_0 + q\vec{\omega})} e^{-\tau(\mathbf{x}_0 + q\vec{\omega}, \mathbf{x}_0 + t\vec{\omega})} \\ \quad = e^{-\tau(\mathbf{x}_0 + s\vec{\omega}, \mathbf{x}_0 + t\vec{\omega})} \end{cases} . \tag{8}$$