*Volume 0 (1981), Number 0 pp. 1–12* COMPUTER GRAPHICS *forum*

# Free Path Sampling in High Resolution Inhomogeneous Participating Media

László Szirmay-Kalos, Balázs Tóth, and Milán Magdics

Budapest University of Technology and Economics, Hungary

**Abstract**

*This paper presents efficient algorithms for free path sampling in heterogeneous participating media defined either by high-resolution voxel arrays or generated procedurally. The method is based on the concept of mixing "virtual" material or particles to the medium, augmenting the extinction coefficient to a function for which the free path can be sampled in a straightforward way. The virtual material is selected such that it modifies the volume density but does not alter the radiance. We define the total extinction coefficient of the real and virtual particles by a low-resolution grid of super-voxels that are much larger than the real voxels defining the medium. The computational complexity of the proposed method depends just on the resolution of the super-voxel grid, and does not grow with the resolution above the scale of super-voxels. The method is particularly efficient to render large, low-density, heterogeneous volumes, which should otherwise be defined by enormously high resolution voxel grids, and where the average free path length would cross many voxels.*

## 1. Introduction

Simulating multiple scattering and rendering inhomogeneous participating media in a realistic way are challenging problems [RT87, Rus94, LBC95, CPP*05, Fat09]. The most accurate approaches are based on Monte Carlo quadrature and trace photons or importons (i.e. visibility rays) randomly in the medium [JC98, PKK00, QXFN07] (Figure 1).

Generating a single step of the random path involves the sampling of the *free path* traveled by the photon before scattering, deciding whether or not absorbtion happens, and finally sampling the new scattering direction. Absorbtion handling and the determination of the new scattering direction are based on the local properties of the medium, thus these tasks can be solved by evaluating relatively simple formulae that need just a few variables, which are either globally available or can be determined from the actual position.

Free path sampling is also simple when the medium is homogeneous since the next scattering point will depend just on the constant *extinction coefficient* of the medium. However, in inhomogeneous media having position dependent density, free path sampling should gather information about the continuously changing extinction coefficient along the way of the photon [CSI09].

A popular method for exploring the volume to decide the position of scattering is *ray marching* that takes small steps along the ray and assumes that between steps the density is constant. For participating media defined by voxel arrays, a safe step size is comparable to the edge length of the voxel. Ray marching should take a lot of steps until collision if the average density is low, i.e. the expected free path is long, and the density has high frequency variations or the voxel array has high resolution, i.e. the steps must be small to al-

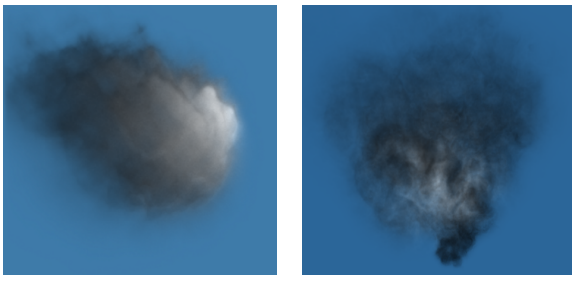


**Figure 1:** *Monte Carlo global illumination rendering of inhomogeneous participating media of* $4096^3$ *effective resolution. The cloud is rendered at 1.2 seconds, the smoke at 8 seconds on an NVIDIA GeForce 480 GPU.*


Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

low the assumption on constant density between them. In high resolution voxel arrays, ray marching will dominate the rendering time.

Another possibility for free path sampling is *Woodcock tracking* that advances in the media with random length steps instead of the constant length steps of ray marching. The expected length of the random steps is determined by the maximum extinction coefficient of the volume. Conceptually, Woodcock tracking is similar to Russian roulette in the sense that both of them randomly decide whether or not a complicated operation is executed and then scale the result in order to compensate those cases when the random decision does not require the execution of the computation. Woodcock tracking offers free path sampling with correct expected values and randomly varying number of steps. As the length of random steps is governed by the maximum extinction coefficient, the probability that we save computation time with respect to ray marching is high if the varying extinction coefficient is close to the maximum value. However, in cases when the extinction is much higher in a small portion of the volume than in the rest, Woodcock tracking becomes prohibitively inefficient.

Our objective is to generalize Woodcock tracking and make it efficient even for participating media where the maximum extinction is far from the extinction values of most points. This paper proposes an efficient free path sampling method for volumes where we have not only the density values of the voxels, but also an upper-bounding function for which free path sampling is straightforward. If the volume is available as a high resolution voxel array, then the bounding function can be computed before rendering starts. In case of procedurally generated volumes [EMP*03], the bounding function can be obtained directly from the procedural definition, without generating the high resolution and therefore very large voxel array.

The paper is structured in the following way. In Section 2 we survey the previous work on free path sampling. Section 3 presents our new method. In Section 4 the method is applied for voxel data and volumes generated procedurally with multi-scale noise. Section 5 discusses a photon mapping global illumination renderer that incorporates the proposed free path sampling algorithms. This system has been implemented in CUDA. Finally, Section 6 presents results and performance evaluation as well.

## 2. Light attenuation and free path sampling

In participating media radiance $L$ is attenuated along ray $\vec{p}(s) = \vec{p}_{start} + \vec{\omega}s$ of origin $\vec{p}_{start}$ and direction $\vec{\omega}$ by absorbtion and out-scattering, which results in an exponential decay:

$$L(s) = L(0) \cdot \exp\left( -\int_0^s \sigma_t(\vec{p}(s'))\mathrm{d}s' \right) \qquad (1)$$

where $\sigma_t(\vec{p})$ is the *extinction coefficient* that defines the probability density that photon-particle collision happens in point $\vec{p}$ provided that the photon arrived at this point.

The integral of the extinction coefficient is also called the *optical depth* of this interval and is denoted by $\tau$:

$$\tau(s_0, s_1) = \int_{s_0}^{s_1} \sigma_t(\vec{p}(s'))\mathrm{d}s'. \qquad (2)$$

Monte Carlo methods generate discrete light path samples and approximate integral quadratures by the weighted sum of the contribution of these paths. The error of the approximation can be reduced by *importance sampling* that places the discrete samples with a frequency that is proportional to the integrand. Sampling proportionally to a prescribed function can be done by the *inversion method*. The inversion method first calculates the probability density as the normalization of the original function, then obtains the desired *cumulative probability distribution* (*CDF*) as the integral of the probability density, and finally generates the discrete samples by inverting the CDF for values that are uniformly distributed in the unit interval.

The CDF of the free path length $s$ along ray $\vec{p}(s)$ is

$$P(s) = 1 - \exp(-\tau(0,s)). \qquad (3)$$

Thus, free path length $s$ corresponding to a uniform random number $r$ is the solution of the following equation:

$$r = P(s) \quad \Leftrightarrow \quad -\log(1-r) = \tau(0,s). \qquad (4)$$

When the medium is inhomogeneous, the extinction coefficient is not constant but is represented by a voxel grid or by other finite elements. In this case, the usual approach is *ray marching* that takes small steps $\Delta s$ along the ray and checks when the Riemann sum approximation of the optical depth gets larger than $-\log(1-r)$:

$$\sum_{i=0}^{n-1} \sigma_t(\vec{p}(i\Delta s))\Delta s \le -\log(1-r) < \sum_{i=0}^{n} \sigma_t(\vec{p}(i\Delta s))\Delta s. \qquad (5)$$

Unfortunately, this algorithm is biased [RSK08] and requires a lot of voxel array fetches, especially when the voxel array is large and the average extinction is small.

*Woodcock tracking* [WMHL65] (also called as fictitious interaction tracking, pseudo-scattering, hole-tracking, self-scattering or delta-tracking) provides an unbiased alternative to *ray marching*, and samples the free path with the following randomized algorithm:

1. Generate tentative path length $s$ using the maximum extinction coefficient $\sigma_{\max}$ in the volume.
2. Accept the tentative collision point with probability $\sigma_t(\vec{p}(s))/\sigma_{\max}$.
3. If the collision is rejected, then the particle's direction is not altered and a similar sampling step is repeated from the tentative collision point.

We do not repeat the proof [Col68] that this method is unbiased here since our new sampling strategy will also contain Woodcock tracking as a special case, and we shall prove the unbiasedness for the general case.

Woodcock tracking becomes very inefficient when the maximum extinction coefficient is much larger than the extinction coefficient in a particular domain of the volume, since here the acceptance probability, $\sigma_t(\vec{p}(s))/\sigma_{\max}$, will be very small, which requires the sampling of a lot of tentative scattering points [Lep10]. In the *Hole geometry package* [Ans], the application developer should decompose the geometry to roughly homogeneous regions where Woodcock tracking can be executed with different maximum extinction parameters. However, this approach requires the inclusion of a fictitious scattering at each crossed boundary of different regions, where a new random sample needs to be generated and the sample process repeated, which degrades performance when the path crosses many regions. In [SKTMC10] Woodcock tracking has been extended to piece-wise constant upper-bounding functions. In this paper we further generalize Woodcock tracking for arbitrary upper-bounding functions that are obtained either during pre-processing or on-the-fly taking advantage of the procedural definition.

## 3. The new method

Free path sampling is equivalent to the solution of equation (4) for path length $s$. If the extinction coefficient and consequently the optical depth are available in a simple algebraic form defined by a few parameters, then the solution is straightforward and requires just the fetching of these parameters from the main memory. However, if the optical depth can only be computed from many data, which happens when the extinction coefficient is specified by a high-resolution voxel array, then the sampling process will be slow. To solve this problem, we modify the volume by adding virtual "material" or particles in a way that the total density will follow a simple function. One might think that modifying the material density would also change the light radiance inside the volume resulting in a distorted rendering solution, which is obviously not desired. Fortunately, this is not necessarily the case if the other two free properties of the virtual material, namely the albedo and the phase function are appropriately defined. Virtual particles do not alter the radiance inside the medium if they do not change the energy and the direction of photons during scattering. This requirement is met if the virtual particle has *albedo* 1, and its *phase function* is a Dirac-delta, since in this case the collision with a virtual particle alters neither the photon's energy nor its direction with probability 1, so the virtual material does not affect the light's radiance (Figure 2).

More formally, we handle heterogeneous volumes by mixing additional *virtual particles* into the medium to augment the extinction coefficient to a simpler upper-bounding function $\sigma_{\max}(\vec{p})$. For original extinction coefficient $\sigma_t(\vec{p})$,

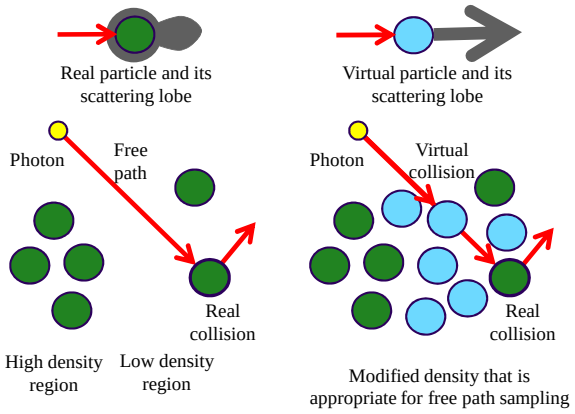



**Figure 2:** *Virtual particles modify the density but not the radiance since their albedo is 1 and their phase function is a Dirac-delta.*

we have to find the extinction coefficient $\sigma_v(\vec{p})$ of the virtual particles, so that in the *combined medium* of real and virtual particles the extinction coefficient is $\sigma_{\max}(\vec{p}) = \sigma_t(\vec{p}) + \sigma_v(\vec{p})$. During scattering we have to determine whether it happened on a real or on a virtual particle. As sampling is required to generate random points with a prescribed probability density, it is enough to solve this problem randomly with the proper probabilities. As the extinction parameters define the probability density of scattering, ratios $\sigma_t(\vec{p})/\sigma_{\max}(\vec{p})$ and $\sigma_v(\vec{p})/\sigma_{\max}(\vec{p})$ give us the probabilities whether scattering happened on a real or on a virtual particle, respectively.

### 3.1. Free path sampling with virtual particles

Having added the virtual particles, free path sampling is executed in the following steps:

1. Generate path length $s$ using the upper-bounding extinction coefficient function $\sigma_{\max}(\vec{p}(s))$.
2. When a potential scattering point $\vec{p}$ is identified, we decide randomly with probability $\sigma_t(\vec{p})/\sigma_{\max}(\vec{p})$ whether scattering happened on a real or on a virtual particle. If only virtual scattering occurred, then the particle's direction is not altered and a similar sampling step is repeated from the scattering point.

Note that this algorithm is quite similar to that of Woodcock tracking. The key difference is that we do not require the maximum extinction coefficient be a global constant, but allow arbitrary non-negative virtual particle density $\sigma_v(\vec{p})$.

### 3.2. Piece-wise polynomial upper-bound

Free path length $s$ in the combined medium of the real and virtual particles is obtained by solving the following equation, which is the adaptation of equation (4) for the medium

containing both the real and the virtual particles:

$$-\log(1-r) = \int_0^s \sigma_{\max}(\vec{p}(s'))\mathrm{d}s'. \qquad (6)$$

If we have a simple representation for $\sigma_{\max}(\vec{p})$, then the scattering point in the combined medium can be found in a simpler way than identifying the scattering point of the medium containing only the real particles.

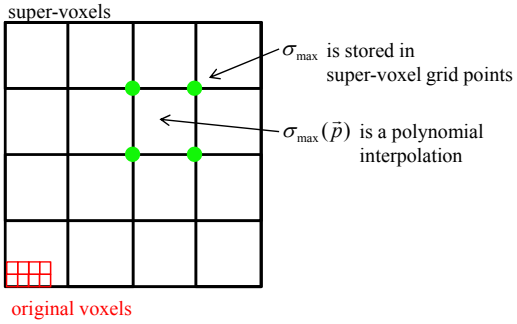



**Figure 3:** *Super-voxel grid that encodes upper-bounding density* $\sigma_{\max}(\vec{p})$.

In this paper we assume that the upper-bounding $\sigma_{\max}(\vec{p})$ is a piece-wise polynomial function defined in a low-resolution grid. The voxels of this low-resolution grid are much larger than the actual voxels defining the true extinction coefficient $\sigma_t(\vec{p})$, and are called *super-voxels* (Figure 3). We note that other upper-bounding functions might also be used if the solution of equation (6) is straightforward for them.

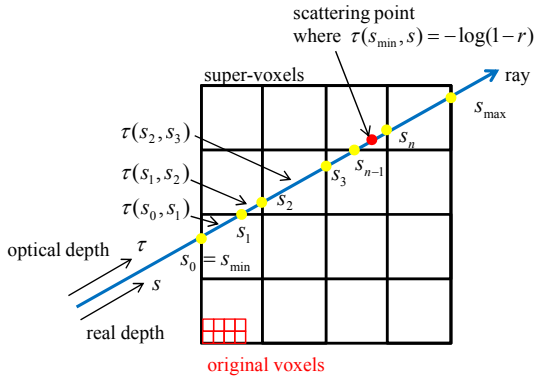



**Figure 4:** *3D DDA algorithm to visit super-voxels.*

We execute a 3D DDA like voxel traversal [FTK86, AW87] on the super-voxels and find the super-voxel that contains the root of equation (6) (Figure 4). The 3D DDA algorithm is based on the recognition that the boundaries of the cells are on three collections of parallel planes, which are orthogonal to the $x$, $y$, and $z$ axes, respectively. The algorithm maintains three ray parameters representing the next intersection points with these plane collections. The minimum of the three ray parameters represents the exit point of the current cell. To step onto the next cell, an increment is added to this ray parameter. The increments corresponding to the three plane collections are constants for a given ray.

As super-voxels are visited one after the other, we check whether the root of equation (6) is in this super-voxel. The inequalities selecting the super-voxel $n$ that contains the scattering point are:

$$\sum_{i=0}^{n-2} \tau_{\max}(s_i, s_{i+1}) \le -\log(1-r) < \sum_{i=0}^{n-1} \tau_{\max}(s_i, s_{i+1}) \qquad (7)$$

where

$$\tau_{\max}(s_i, s_{i+1}) = \int_{s_i}^{s_{i+1}} \sigma_{\max}(\vec{p}(s'))\mathrm{d}s'$$

is the *optical depth* of the ray segment intersecting the $i$th super-voxel, computed for the upper-bounding extinction coefficient.

The important differences of ray marching and the proposed approach are that steps $\Delta s_i = s_{i+1} - s_i$ are not constant but are obtained as the length of the intersection of the ray and the super-voxel, and sample points $s_i$ are consequently on super-voxel boundaries.

When in step $n$ the inequalities are first satisfied, the super-voxel of the scattering point is located. The actual scattering point is computed by solving the following equation for $s$:

$$\tau(s_{n-1}, s) = -\log(1-r) - \sum_{i=0}^{n-2} \tau_{\max}(s_i, s_{i+1}) \qquad (8)$$

where the right side is computed from the uniform sample $r$ and the optical depth of previously visited super-voxels.

To find the super-voxel containing the scattering point and to locate the scattering point inside this super-voxel, we need to compute the optical depth along the ray segments intersecting the super-voxels. Let us consider a single super-voxel and use normalized coordinates $X, Y, Z$ that are all zeros at the left-front-lower corner of the super-voxel and all ones at its right-back-upper corner (Figure 5). The optical depth integral of a single super-voxel can be computed from the extinction coefficients $\sigma_{000}, \ldots, \sigma_{111}$ of the corners, where the first, second, and third indices denote the $X, Y, Z$ coordinates, respectively.

The ray enters the super-voxel at *entry point* $\vec{e}$ and leaves it at *exit point* $\vec{o}$. Points $\vec{p}$ of the ray segment that are inside the super-voxel satisfy the following equation:

$$\vec{p}(s) = (1 - t(s))\vec{e} + t(s)\vec{o}$$

where $t(s) = (s - s_i)/\Delta s_i \in [0, 1]$. As extinction coefficient

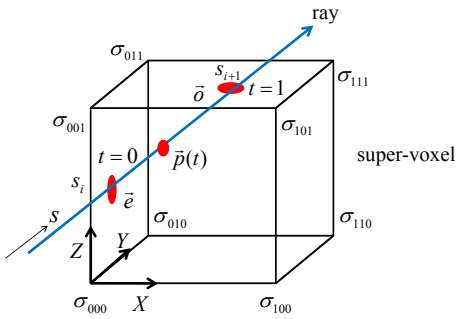



**Figure 5:** *Processing of a single super-voxel. The ray enters the super-voxel at entry point $\vec{e}$ and leaves it at exit point $\vec{o}$. The extinction coefficient equals to $\sigma_{000},\ldots,\sigma_{111}$ at the corner points, and is a polynomial of ray parameter $t$ inside the super-voxel.*

```
SuperVoxelTraversal( p_start, ω ⇒ p)
    do                                  // Loop until a real scattering is found
        τ_sample = −log(1 − rnd());     // Sample's optical depth
        o = p_start; s_o = 0; τ_o = 0;                  // Exit point
        while τ_o < τ_sample do                         // 3D DDA loop
            e = o; s_e = s_o; τ_e = τ_o;                // Entry point
            s_o = Ray parameter of the next intersection;
            if (out of volume) return "No scattering";
            o = r + s_o ω;                              // Next exit point
            PolyCoeff(e, o ⇒ c_0, ..., c_M);            // Polynomial
            τ_o = τ_e + (s_o − s_e) · Σ_{d=0}^{M} c_d/(d+1);
        endwhile
        SolvePolyInt(c_0, ..., c_M, s_e, s_o, τ_e, τ_o, τ_sample ⇒ s, t);
        σ_max = Σ_{d=0}^{M} c_d t^d;                    // Max extinction coefficient
        p = p_start + sω;                               // Next scattering point
        P_real = σ_t(p)/σ_max;                          // Probability of real scattering
    while (rnd() > P_real);                             // Is real or virtual?
    return "Scattering";                                // Real scattering is found
end
```

$\sigma_{\max}(\vec{p})$ is a tri-variate polynomial of the Cartesian coordinates, which are linear functions of parameter $t$, the extinction coefficient will be a polynomial of parameter $t$:

$$\sigma_{\max}(\vec{p}(s(t))) = \sum_{d=0}^{M} c_d t^d, \tag{9}$$

where $(c_0,\ldots,c_M)$ are the coefficients of the polynomial. Thus, the optical depth between entering this super-voxel and parameter $s$ is:

$$\tau_{\max}(s_i,s) = \int_{s_i}^{s} \sigma_{\max}(\vec{p}(s'))\mathrm{d}s' =$$

$$\Delta s_i \int_0^t \sigma_{\max}(\vec{p}(s(t')))\mathrm{d}t' = \Delta s_i \sum_{d=0}^{M} c_d \frac{t^{d+1}}{d+1}. \tag{10}$$

The total optical depth $\tau(s_i,s_{i+1})$ in super-voxel $i$ can be obtained by substituting the end point, i.e. $s=s_{i+1}$ or $t=1$:

$$\tau_{\max}(s_i,s_{i+1}) = \Delta s_i \sum_{d=0}^{M} \frac{c_d}{d+1}. \tag{11}$$

When the super-voxel that contains the scattering point is identified, the root of equation (6) can be obtained by the false position root finding method since the function is monotonically increasing.

### 3.3. Algorithmic details

The algorithm to find the next scattering point $\vec{p}$ taking the ray of origin $\vec{p}_{start}$ and direction $\vec{\omega}$ is a nested loop. The outer loop is responsible for random decisions whether a virtual or a real particle is hit. The inner loop is a 3D DDA voxel traversal of the super-voxel grid.

The outer loop starts with sampling the optical depth $\tau_{sample}$ by transforming a random value uniformly distributed in the unit interval, generated by the rnd() function. Then, the inner loop visits super-voxels, maintaining entry point $\vec{e}$ with its ray parameter $s_{\vec{e}}$ and optical depth $\tau_{\vec{e}}$, as well as exit point $\vec{o}$ with its ray parameter $s_{\vec{o}}$ and optical depth $\tau_{\vec{o}}$. Initially, the entry point is the origin of the ray with zero ray parameter and optical depth, then the parameters of the entry point are updated with the parameters of the exit point of the previous super-voxel at each DDA step. The ray parameter of the exit point $s_{\vec{o}}$ is determined by the 3D DDA algorithm, and the location of the exit point is obtained by inserting the ray parameter into the equation of the ray. Function **PolyCoeff** computes the coefficients of the polynomial of the upper-bounding optical depth in this super-voxel. The optical depth of the exit point is computed as the sum of that of the entry point and the integral of the polynomial in this super-voxel. The inner loop is executed until the optical depth of the exit point becomes larger than the sampled optical depth, i.e. when the super-voxel contains the sample point. Exiting the inner loop, we have to identify the exact location of the scattering point, which is the solution of the sampling equation by calling **SolvePolyInt**, which returns ray parameter $s$ and its normalized version $t$. The upper-bounding extinction coefficient $\sigma_{\max}$ is computed by substituting the ray parameter into the polynomial, and the location of the scattering by inserting the ray parameter into the equation of the ray. At the end of the outer loop, we decide randomly whether this point corresponds to a real or a virtual scattering having fetched extinction $\sigma_t(\vec{p})$ from the original volume.

The computation of the polynomial coefficients by **PolyCoeff** and the solution of the sampling equation by **SolvePolyInt** depend on the actual form of the upper-bounding function. In the following subsections, we provide the implementations for the piece-wise constant and tri-linear cases.

#### 3.3.1. Piece-wise constant upper-bound

If the upper-bounding $\sigma_{\max}(\vec{p})$ is constant in a super-voxel, then $c_0 = \sigma_{\max}(\vec{p})$ in this super-voxel and all other coefficients are zero. The optical depth will be a linear function of ray parameter $t$, thus equation $\tau(s_i, s) = \tau_{sample} - \tau_{\vec{e}}$ can be solved directly:

**SolvePolyInt**$(c_0, s_{\vec{e}}, s_{\vec{o}}, \tau_{\vec{e}}, \tau_{\vec{o}}, \tau_{sample} \Rightarrow s, t)$
$t = (\tau_{sample} - \tau_{\vec{e}})/(s_{\vec{o}} - s_{\vec{e}})/c_0;$
$s = s_{\vec{e}} + (s_{\vec{o}} - s_{\vec{e}}) \cdot t;$
**end**

#### 3.3.2. Tri-linearly interpolated upper-bound

In the case of tri-linear upper-bound, the polynomial of $\sigma_{\max}(\vec{p}(t))$ is cubic with coefficients $(c_0, c_1, c_2, c_3)$. The coefficients are computed from the extinction values $\sigma_{000}, \ldots, \sigma_{111}$ in the eight corners and from the end points of the ray segment. First, we transform the entry and exit points into normalized coordinates, where the left-front-lower corner of the super-voxel is the origin and the right-back-upper corner has coordinates (1,1,1), resulting in transformed entry point $\vec{E}$ and exit point $\vec{O}$. In this space, the extinction coefficient is the following function of normalized coordinates $X, Y, Z$:

$$\sigma_{\max}(X,Y,Z) = \sigma_{000}\bar{X}\bar{Y}\bar{Z} + \sigma_{100}X\bar{Y}\bar{Z} + \sigma_{010}\bar{X}Y\bar{Z} +$$

$$\sigma_{001}\bar{X}\bar{Y}Z + \sigma_{110}XY\bar{Z} + \sigma_{101}X\bar{Y}Z + \sigma_{011}\bar{X}YZ + \sigma_{111}XYZ \quad (12)$$

where we used the shorthand notation $\bar{A} = 1 - A$. Substituting the ray of equation $(X(t), Y(t), Z(t)) = (1-t)\vec{E} + t\vec{O}$ into this equation, the maximum coefficient is expressed as a cubic polynomial of $t$, where the similar powers of $t$ are grouped and the coefficients are expressed. The computation is summarized in the following function:

**PolyCoeff**$(\vec{e}, \vec{o} \Rightarrow c_0, c_1, c_2, c_3)$
$\vec{E}$ = Transform entry point $\vec{e}$ to the unit voxel cube;
$\vec{O}$ = Transform exit point $\vec{e}$ to the unit voxel cube;
$(\Delta x, \Delta y, \Delta z) = (\vec{O}_x - \vec{E}_x, \vec{O}_y - \vec{E}_y, \vec{O}_z - \vec{E}_z);$
$d_{xyz} = \sigma_{111} - \sigma_{011} - \sigma_{101} - \sigma_{110} + \sigma_{100} + \sigma_{010} + \sigma_{001} - \sigma_{000};$
$d_{xy} = \sigma_{000} - \sigma_{100} - \sigma_{010} + \sigma_{110};$
$d_{xz} = \sigma_{000} - \sigma_{100} - \sigma_{001} + \sigma_{101};$
$d_{yz} = \sigma_{000} - \sigma_{010} - \sigma_{001} + \sigma_{011};$
$d_x = \sigma_{100} - \sigma_{000};\ d_y = \sigma_{010} - \sigma_{000};\ d_z = \sigma_{001} - \sigma_{000};$
$c_3 = d_{xyz}\Delta x\Delta y\Delta z;$
$c_2 = (\vec{E}_z\Delta x\Delta y + \vec{E}_y\Delta x\Delta z + \vec{E}_x\Delta y\Delta z)d_{xyz} +$
$d_{yz}\Delta x\Delta y + d_{xz}\Delta x\Delta z + d_{yz}\Delta y\Delta z;$
$c_1 = (\vec{E}_x\vec{E}_z\Delta x + \vec{E}_x\vec{E}_z\Delta y + \vec{E}_x\vec{E}_y\Delta z)d_{xyz} + d_x\Delta x + d_y\Delta y + d_z\Delta z +$
$(\vec{e}_y\Delta x + \vec{e}_x\Delta y)d_{xy} + (\vec{E}_z\Delta x + \vec{E}_x\Delta z)d_{xz} + (\vec{E}_z\Delta y + \vec{E}_y\Delta z)d_{yz};$
$c_0 = \vec{E}_x\vec{E}_y\vec{E}_z d_{xyz} + \vec{E}_x\vec{E}_y d_{xy} + \vec{E}_x\vec{E}_z d_{xz} + \vec{E}_y\vec{E}_z d_{yz} +$
$\vec{E}_x d_x + \vec{E}_y d_y + \vec{E}_z d_z + \sigma_{000};$
**end**

Function **SolvePolyInt** finds the solution of $\tau(s_i, s) = \tau_{sample} - \tau_{\vec{e}}$ by the *false position method* (regula falsi):

**SolvePolyInt**$(c_0, \ldots, c_M, s_{\vec{e}}, s_{\vec{o}}, \tau_{\vec{e}}, \tau_{\vec{o}}, \tau_{sample} \Rightarrow s, t)$
$t_{low} = 0; \quad t_{high} = 1;$
$\tau_{low} = \tau_{\vec{e}}; \quad \tau_{high} = \tau_{\vec{o}};$
**while** $(t_{high} - t_{low} >$ EPS)
$t = t_{low} + (t_{high} - t_{low}) \cdot (\tau_{sample} - \tau_{low})/(\tau_{high} - \tau_{low});$
$\tau = \tau_{\vec{e}} + (s_{\vec{e}} - s_{\vec{o}}) \cdot \sum_{d=0}^{M} c_d t^{d+1}/(d+1);$
**if** $(\tau < \tau_{sample})$ $t_{low} = t$, $\tau_{low} = \tau;$
**else** $t_{high} = t$, $\tau_{high} = \tau;$
**endwhile**
$s = (s_{\vec{o}} - s_{\vec{e}}) \cdot t + s_{\vec{e}};$
**end**

### 3.4. Attenuation calculation with virtual particles

In both physically based and simplified simulations, we often have to determine the total radiance attenuation between two points. If scattering is ignored, the total attenuation between the source and the camera will determine the image. In volume shadowing, the attenuation between the source and the current sample point is needed to find the shadowing factor. In Monte Carlo simulation, paths are usually completed by deterministic connections [DBB03], and the attenuation along this connection is included into the estimator. If the path is started in the source (photon tracing), then scattering points are connected to the eye. Alternatively, in path tracing, scattering points are usually connected to the sources.

In all these calculations the integral of the optical depth should be computed for the line segment. The optical depth can be obtained by ray marching at the expense of many texture fetches. Another possibility is the direct application of the proposed free path sampling. We decide randomly whether or not the free path is longer than the line segment. If the random free path is longer, the radiance at the start point is added to the end without attenuation. If the random free path is shorter, then no radiance is added. Although the expected value of this estimator provides correct results, its variance is too high.

Thus, we apply the concept of virtual particles in a better way. The optical depth is decomposed to two terms corresponding to the upper-bounding extinction and the negative extinction of the virtual particles:

$$\int_0^s \sigma_t(\vec{p}(s'))\mathrm{d}s' = \int_0^s \sigma_{\max}(\vec{p}(s'))\mathrm{d}s' - \int_0^s \sigma_v(\vec{p}(s'))\mathrm{d}s'. \quad (13)$$

The integral of the upper-bounding extinction can be analytically computed. The integral of the extinction in virtual particles is estimated by a numerical quadrature. If the difference of the maximum and the actual extinction coefficients (the extinction of the virtual particles) has relatively low variation, then a few samples can provide acceptable accu-

racy. This idea, its advantages and drawbacks are similar to those of the *separation of the main part* used in Monte Carlo approaches. If the upper-bounding extinction coefficient is tight, then the estimation error can be greatly reduced. However, if the upper-bound is a very crude approximation and is farther from the extinction coefficient than the constant zero function, then this method is not worth using.

## 4. Obtaining the upper-bounding extinction function

In this section we consider two volume representations including voxel arrays and procedurally defined participating media, and discuss how the upper-bounding extinction function can be obtained for them.

### 4.1. Voxel arrays

Measured or computed volumes are often defined in a discretized form, as a 3D regular grid of voxel values, which can have large resolution. If we use ray marching, then the computational complexity of free path sampling will be linear in terms of the linear voxel grid resolution. In order to use the proposed method, we need an upper-bounding functional representation defined on a low-resolution grid. An upper-bounding zero-order polynomial, i.e. a single maximum value of voxels in each super-voxel is easy to find. For tri-linear or higher-order functional upper-bounds, we use the following algorithm. First, the original voxel values are considered only at the super-voxel corners and a function is fitted to these samples. Then, the original voxel values inside each super-voxel are compared to the initial functional representation, and the maximum difference between them is obtained. Finally, this maximum difference is added to the functional upper-bound. Note that using this method, neighboring super-voxels may have different values at the shared super-voxel corners. The increased storage is small since the super-voxel grid has a low-resolution. However, should it pose problems, the maximum value at each shared corner is computed, and assigned to the corner as the value of every super-voxel sharing it.

### 4.2. Procedural volumes

Volumes are collections of small particles of given cross sections and varying densities. Thinking of the forms of a cloud, fire or smoke, we note that these phenomena have some general smooth shapes, which are perturbed by random and high frequency fluctuations. Thus, at point $\vec{p}$ the varying density and consequently extinction coefficient $\sigma_t(\vec{p})$ are defined by smooth *shape function* $F(\vec{p})$ that is randomly perturbed by *noise* $n(\vec{p})$. Here *perturbation* may mean, for example, addition ($\sigma_t(\vec{p}) = F(\vec{p}) + n(\vec{p})$), multiplication ($\sigma_t(\vec{p}) = F(\vec{p}) \cdot n(\vec{p})$), the translation of variable $\vec{p}$ by a random offset ($\sigma_t(\vec{p}) = F(\vec{p} + \vec{n}(\vec{p}))$), or any combination of these.

Natural noise has fractal characteristics, or from another point of view, its power density function decreases at higher frequencies unlike that of the white noise, which delivers the same power on all frequencies. To create a *fractal noise*, we can add up elemental noise functions defined on different scales. On a given scale, *elemental noise function* $\nu(\vec{p})$ is defined on a 3D integer grid and smoothed in between the grid points with separable interpolating filters. In *value noise* [Lew89] the values are specified randomly in grid points. *Gradient noise* [Per85] assigns random gradients or planes to the grid points, evaluates the signed distance between these planes and the point of interest, and interpolates the distances. *Wavelet noise* [CD05] is generated by an algorithm similar to that of value noise but obtains scales as different smoothed versions of the same white noise. All these noise construction methods result in a tri-variate polynomial of Cartesian coordinates $X, Y, Z$ inside a voxel, where the coefficients can be computed from the given values or gradients in the grid points. In the simplest case of tri-linearly interpolated value noise, the polynomial is a tri-linear function.

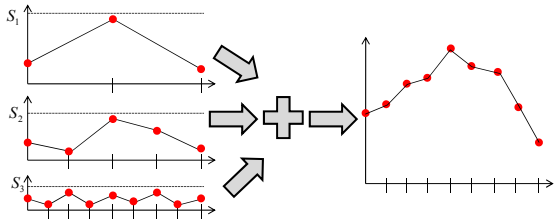



**Figure 6:** *1D value noise defined as the sum of interpolated white noise functions of different scales.*

From the perspective of our proposed method, it is important to note that the elemental noise function can easily be lower-bounded and upper-bounded between grid points since the tri-linear interpolating filters are monotonic. For value noise, the minimum and the maximum of the function inside the grid cell are the minimum and maximum of the values assigned to the corners of the cell. As random values of interval [0,1) are assigned to the corners, we obtain for the minimum and the maximum $\nu_{\min} = 0$ and $\nu_{\max} = 1$, respectively. For gradient noise, a signed distance is calculated for the sampling point, which is then interpolated. The signed distance is in $[-\sqrt{3}, \sqrt{3}]$ in a unit sized cube and tri-linear interpolation also keeps the resulting value in this interval, thus $\nu_{\min} = -\sqrt{3}$ and $\nu_{\max} = \sqrt{3}$.

In a multi-scale noise representation, the linear grid resolution is doubled at each scale, introducing new "octaves". Meanwhile, the amplitude is decreased at higher scales, which is responsible for the $1/f$-noise characteristics. Gen-

erally, a multi-scale noise function is

$$n(\vec{p}) = \sum_{l=1}^{N} S^{(l)} \nu(2^l \vec{p}) \tag{14}$$

where $S^{(l)}$ defines the amplitude at scale $l$. A typical selection of the amplitudes is $S^{(l)} = 1/2^l$. Fractal noise was successfully applied to static 3D volumes and animated 3D, also called 4D volumes, including clouds, fire, and other natural phenomena [Per85].

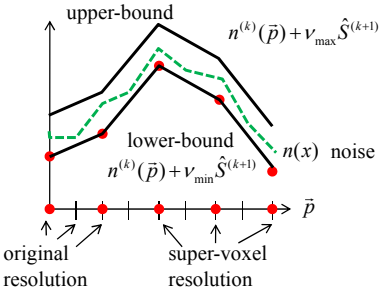



**Figure 7:** *At level k the multi-scale noise can be enclosed by two piece-wise polynomial functions defined on the super-voxel grid. The distance between the upper and lower bounds depends on the amplitudes of the ignored scales.*

If we replace the scales above $k$ by the bounds, the multi-scale noise can also be lower-bounded and upper-bounded without generating the finer scales:

$$n^{(k)}(\vec{p}) + \nu_{\min}\hat{S}^{(k+1)} \le n(\vec{p}) \le n^{(k)}(\vec{p}) + \nu_{\max}\hat{S}^{(k+1)} \tag{15}$$

where

$$\hat{S}^{(k+1)} = \sum_{l=k+1}^{N} S^{(l)}$$

is the amplitude sum of scales above $k$.

If we select the scale and the super-voxel grid so that shape function $F$ is smooth enough in a super-voxel to allow its approximation by an upper-bound $F_{\max}$, then the bounding scheme can also be generalized for the extinction coefficient. In the cases of additive and multiplicative perturbations, the upper-bound is obtained with the maximum shape function $F_{\max}$ and the tri-linear upper-bound of the noise, which is also a tri-linear function in the super-voxels. In this paper we examine only additive and multiplicative noise machines, but also note that the more general case of point translation can also be traced back to these simple cases if the gradient of the shape function is upper-bounded and lower-bounded in the super-voxels.

## 5. The complete rendering algorithm

The discussed free path sampling method can be integrated into Monte Carlo algorithms, e.g. photon tracing, path tracing, bi-directional path tracing, or photon mapping. Photon tracing, path tracing, and bi-directional path tracing connect scattering points deterministically to the sources, to the camera, or to already generated scattering points. Note that only real scattering points should be connected, virtual scattering points serve only as an aid to find the real scattering locations.

For demonstration, we implemented a photon mapping application, which decomposes rendering to a shooting and a gathering phase. During shooting, multiple scattering of photons is calculated in the volume, registering the real scattering points, the photon powers, and the incident directions. The products of photon powers and phase functions evaluated for the incident and camera directions are summed in a 3D array of *gathering super-voxels*, thus we save the expensive construction of a kd-tree and also nearest neighbor search. Finally, the volume is rendered from the camera accumulating the attenuation and the radiance in gathering super-voxels that are along the view ray.

In a gathering super-voxel at point $\vec{x}$, the in-scattered radiance $L(\vec{x},\vec{\omega}')$ is obtained from the in-scattered flux [JC98]:

$$L(\vec{x},\vec{\omega}') = \frac{1}{\sigma_t(\vec{x})} \frac{\mathrm{d}^2\Phi(\vec{x},\vec{\omega}')}{\mathrm{d}\omega \mathrm{d}V} \tag{16}$$

where $\Phi(\vec{x},\vec{\omega}')$ is the in-scattered flux from direction $\vec{\omega}'$. The in-scattered flux is obtained from the photon hits stored in the gathering super-voxel. The radiance scattered towards the eye being in direction $\vec{\omega}$ is

$$L^s(\vec{x},\vec{\omega}) = a(\vec{x}) \frac{\sum_i \Delta\Phi_i P(\vec{\omega}_i',\vec{\omega})}{\Delta V} \tag{17}$$

where $a(\vec{x})$ is the albedo of the volume, $P(\vec{\omega}',\vec{\omega})$ is the *phase function*, $\Phi_i$ and $\vec{\omega}_i'$ are the power and the incident direction of the $i$th photon hit, respectively, and $\Delta V$ is the volume of the gathering super-voxel where photon hits are considered.

In medical simulation, we are interested in the radiation dose of the body tissues, which requires another visualization scheme. Instead of the scattered radiance, the power in-scattered from all directions per unit volume is calculated:

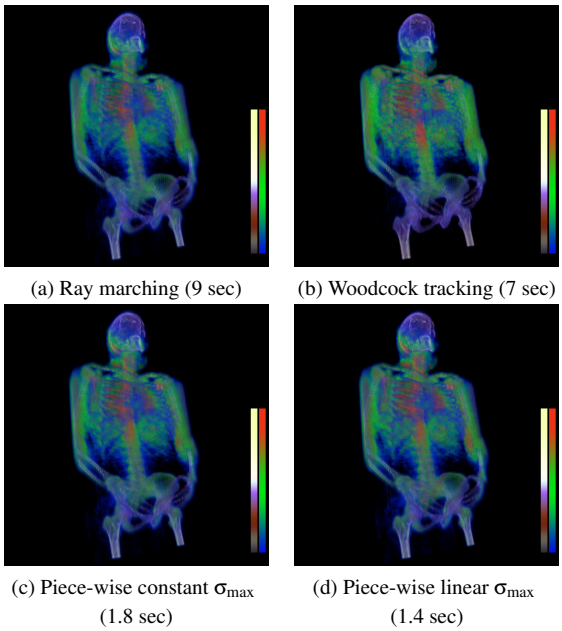
$$I(\vec{x}) = \frac{\sum_i \Delta\Phi_i}{\Delta V} \tag{18}$$

Then, assigning a pseudo-color to the radiation dose by an appropriate transfer function, the volume is rendered with standard alpha blending.

## 6. Results

We demonstrate the proposed free path sampling scheme in two applications: (1) multiple scattering simulation in a high-resolution voxel array and (2) global illumination rendering of procedurally generated participating media. The proposed methods have been implemented in CUDA and their performance measured on an NVIDIA GeForce 480 GPU.

### 6.1. Radiation transfer in voxel arrays



(a) Ray marching (9 sec) (b) Woodcock tracking (7 sec)

(c) Piece-wise constant $\sigma_{max}$ (1.8 sec) (d) Piece-wise linear $\sigma_{max}$ (1.4 sec)

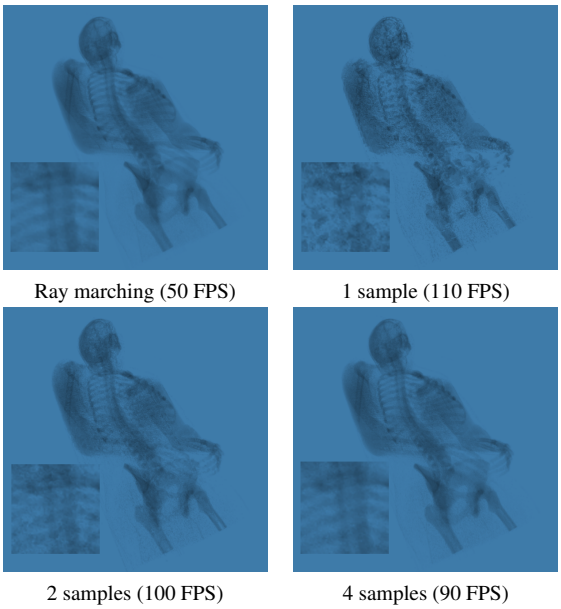
**Figure 8:** *Radiation dose calculation in the Visible Human of* $512^3$ *resolution voxel array. The power density is shown with pseudo-colors obtained with the transfer function of the right bar, and is superimposed onto the density field visualized with the transfer function of the left bar. Times refer to building the photon map containing 32 million hits.*

To measure the performance, we took the standard Visible Human data set and resampled it at $512^3$ resolution to fit it into the GPU memory, simulated multiple scattering and computed the radiation dose with four different free-path sampling methods: (a) classical ray marching, (b) Woodcock tracking, (c) the new method using piece-wise constant upper-bound, and (d) with piece-wise tri-linear upper bound. The resolution of the super-voxel grid was $16^3$. The images obtained with 32 million photons are shown by Figure 8. Note that the visual quality is similar since all methods use the same probability density and the same number of the samples, but the photon shooting times significantly differ since the cost of finding the scattering point is method dependent.

| Upper-bound | $1^3$ | $2^3$ | $4^3$ | $8^3$ | $16^3$ | $32^3$ |
|---|---|---|---|---|---|---|
| Constant | 5 | 10 | 12 | 14 | 18 | 15 |
| Linear | 5 | 11 | 14 | 19 | 22 | 16 |

**Table 1:** *The effect of the super-voxel grid resolution on the performance, which is expressed as million rays per second traced in the Visible Human data set.*

The dependence of the performance on the super-voxel grid resolution is shown by Table 1. At higher super-voxel resolution, rays cross more super-voxels, which requires slightly more computation. On the other hand, increasing the resolution makes the upper-bound tighter, thus it reduces the probability of virtual collisions and consequently the number of texture fetches from the large voxel array. This effect is strong if the extinction coefficient has high variation, thus the optimal tradeoff depends on this factor.



Ray marching (50 FPS) 1 sample (110 FPS)

2 samples (100 FPS) 4 samples (90 FPS)

**Figure 9:** *Attenuation calculation for the Visible Human, with ray marching visiting voxels, and with the proposed scheme taking 1, 2, and 4 samples in each super-voxel. Rendering times are measured at* $600 \times 600$ *image resolution.*

We tested the proposed attenuation calculation algorithm in an X-ray like scenario (Figure 9). Assuming a constant radiance planar source behind the body, the pixel intensity is proportional to the exponential of the optical depth between the source and the camera along the ray of the pixel. Here, the images are obtained by ray marching and by the proposed attenuation estimation method taking 1, 2, or 4 samples in each super-voxel. The step size of ray marching is equal to the linear size of the voxel in the original $512^3$ resolution array. The resolution of the super-voxel grid is $32^3$. Note that the image rendered with as few as 4 samples per super-voxel is very similar to the image computed with more effort due to the fact that the upper-bounding function is a good approximation and is integrated analytically.

### 6.2. Rendering participating media defined by Perlin noise

First we take a lower density variation and a higher density variation cloud model defined by ellipsoids multiplied by 8-octave and 12-octave Perlin noise machines, which correspond to $256^3$ and $4096^3$ effective resolution voxel arrays, respectively. The resolution of the super-voxel grid is $16^3$ in both cases. The scene is illuminated by a single point source. The images of the lower-variation and higher-variation cloud models are shown by Figures 10 and 11. The shooting phase traces 8 million photon rays. The resolutions of the gathering super-voxels are $64^3$ and $256^3$ for low-resolution and high-resolution clouds, respectively.

Table 2 compares the ray shooting performance obtained with classical ray marching (RM), Woodcock tracking, and the new method using piece-wise constant and tri-linear upper-bounds. Note that even at low resolution, the proposed method is faster than ray marching and Woodcock tracking. The advantage of the new method over Woodcock tracking becomes very significant when the volume density has higher variation, and the tri-linear upper-bound adds a 10% speed increase to the method of piece-wise constant upper-bound.

| Method | LV/$256^3$ | HV/$256^3$ | LV/$4096^3$ | HV/$4096^3$ |
|---|---|---|---|---|
| RM | 1.1 | 1.0 | 0.05 | 0.05 |
| Woodcock | 3.3 | 2.0 | 2.2 | 1.2 |
| New/constant | 9.1 | 8.1 | 5.2 | 4.9 |
| New/linear | 10 | 8.9 | 6.3 | 5.7 |

**Table 2:** *Performance expressed as millions of shot rays per second when rendering the lower-variation (LV) and the higher-variation (HV) cloud models generated at both $256^3$ and $4096^3$ resolutions. Note that ray marching (RM) becomes prohibitively slow for high resolution models. Neither Woodcock tracking nor the new method are sensitive to the resolution increase, and the new method outperforms Woodcock tracking even for the lower-variation model when the cloud fills 60% of the complete voxel array. In case of the higher-variation model where clouds occupy about 10% of the volume, Woodcock tracking degrades due to the high number of tentative collisions, but the new method maintains its speed since the piece-wise tri-linear approximation becomes quite accurate in this case.*

When the resolution is higher, ray marching becomes very slow, but Woodcock tracking and the new method are still fast, and the new method keeps its advantage over Woodcock tracking for inhomogeneous volumes (Table 3). The performance advantage of the tri-linear upper-bounding function is about 10% for these models. This factor, similarly to the optimal resolution of the super-voxel grid, is scene dependent.

The image resolution is $1000 \times 1000$ which is equivalent to 1 million gathering rays. We implemented ray marching and the proposed attenuation calculation for gathering. Ray marching and the new method could trace 2.4 million gathering rays per second and 7.2 million rays per second in the $256^3$ resolution volume, respectively. For the higher, $4096^3$ resolution case, the performance of ray marching dropped below 0.1 million rays per second, while the new method still maintained the 3.6 million gathering rays per second speed. With 8 million shooting and 1 million gathering rays, the new method can render multiple scattering effects in a volume of $4096^3$ effective resolution in a second.

Figure 12 shows a frame of a smoke animation sequence where the procedural model is based on 4D Perlin noise. We

| Method | $256^3$ | $512^3$ | $2048^3$ | $8182^3$ | $32768^3$ |
|---|---|---|---|---|---|
| RM | 1.1 | 0.5 | 0.1 | 0.02 | 0.005 |
| Woodcock | 3.3 | 2.9 | 2.5 | 2.0 | 1.6 |
| New/constant | 9.1 | 7.8 | 7.1 | 5.8 | 5.1 |
| New/linear | 10 | 8.5 | 8.0 | 6.0 | 5.5 |

**Table 3:** *Analysis of the scalability. The numbers show the million rays per second with respect to the effective resolution of the cloud model. The new method scales very well and its slight performance decrease is due to the more expensive procedural evaluation of the additional octaves in the extinction coefficient.*

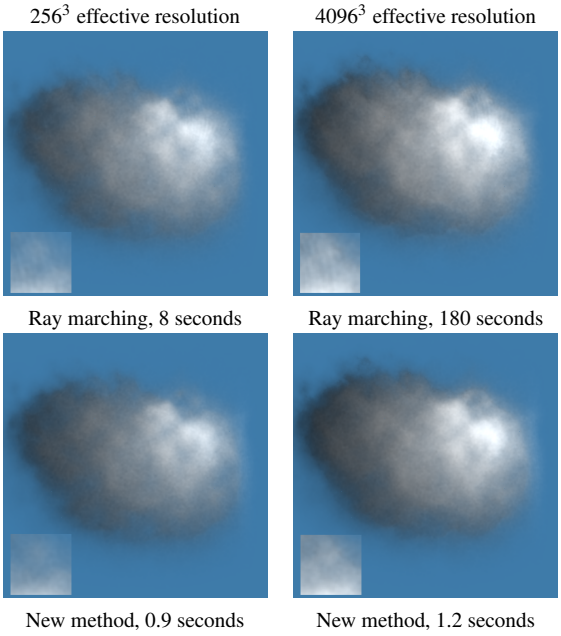



**Figure 10:** *Single cloud, lower-variation models generated at $256^3$ and $4096^3$ effective resolutions. The resolution of the super-voxel grid is $16^3$. We traced 8 million shooting and 1 million gathering rays.*

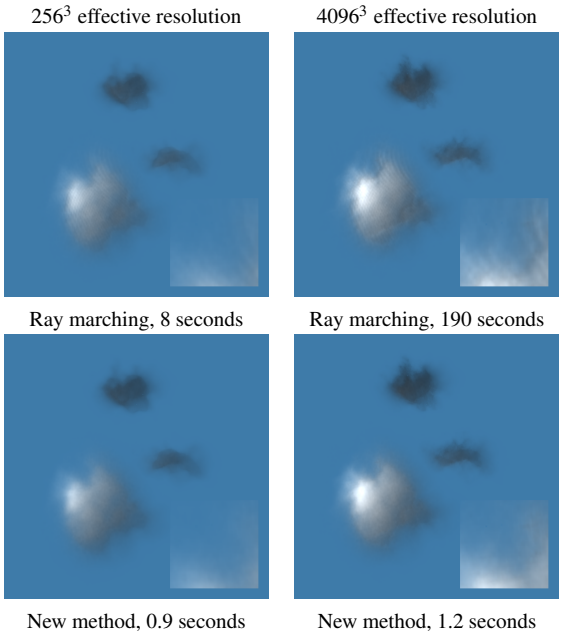



**Figure 11:** *Multi cloud, higher-variation models generated at* $256^3$ *and* $4096^3$ *effective resolutions. The resolution of the super-voxel grid is* $16^3$*. We traced 8 million shooting and 1 million gathering rays.*

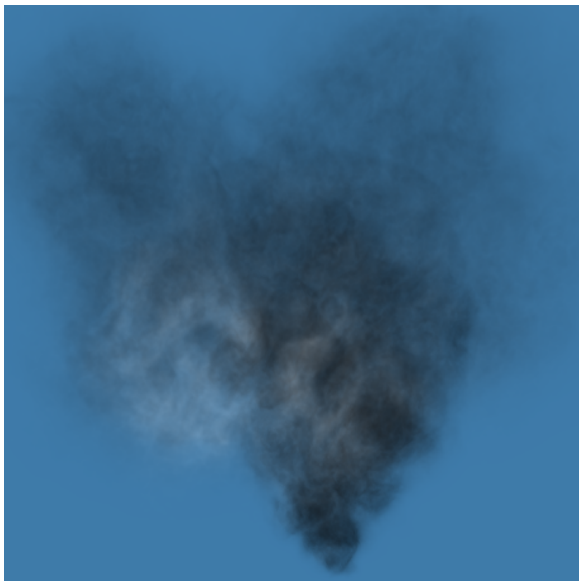


**Figure 12:** *A frame of a smoke animation rendered with multiple scattering. The effective resolution is* $4096^3$*, the super-voxel grid resolution is* $128^3$*.*



defined 12 octaves for the spatial domain, which is equivalent to $4096^3$ effective resolution. Shooting 50 million photons and rendering an image take 8 seconds.

## 7. Conclusions

This paper proposed an effective method to sample the free path length and to compute the optical depth in inhomogeneous participating media, and we used this method to find the global illumination solution of high-resolution voxel arrays or procedurally defined volumetric models. The strength of the method is that its computational complexity depends just on the resolution of the super-voxel data and is independent of the number of additional scales, thus volumes of very large effective resolution can be rendered with multiple scattering effects in seconds. This kind of scalability is due to the randomization of the volume by adding virtual particles and deciding randomly whether a collision happened on these or on real particles. Randomization may have overhead, which is small if the super-voxels absorb the major variations of the original data, and thus the scales above the super-voxels have small amplitude. This is fortunately the case of many natural phenomena having smoother large shapes and small random fluctuations.

Unlike procedurally defined models, measured voxel arrays should be stored into the GPU memory, which limits their resolution. As our proposed method accesses the high resolution array just to decide whether or not a real collision happened, and otherwise uses only the low-resolution array of super-voxels, it is a natural extension to store only the low-resolution array in the GPU memory and copy the real extinction coefficients from the CPU memory only when they are needed. This extension would allow practically arbitrarily high resolution models to be simulated, which is badly needed in medical applications. However, the data access pattern of the high resolution array is very incoherent, so realizing a caching scheme for it in the GPU memory is not simple and is considered as a future work.

## Acknowledgement

This work has been supported by the TeraTomo project of the National Office for Research and Technology, OTKA K-719922, and by the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002). The authors are grateful to NVIDIA for donating the GeForce 480 GPU cards.

## References

[Ans] ANSWERS:. http://www.sercoassurance.com/answers/ resource/ pdfs/hole.pdf.

[AW87] AMANATIDES J., WOO A.: A fast voxel traversal algorithm for ray tracing. In *Proceedings of Eurographics '87* (1987), pp. 3–10.

[CD05] COOK R. L., DEROSE T.: Wavelet noise. In *Computer Graphics (SIGGRAPH '05 Proceedings)* (2005), pp. 803–811.

[Col68] COLEMAN W.: Mathematical verification of a certain monte carlo sampling technique and applications of the technique to radiation transport. *Problems. Nuclear Science and Engineering 32* (1968), 76–81.

[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), 303–328.

[CSI09] CHA D., SON S., IHM I.: GPU-assisted high quality particle rendering. *Computer Graphics Forum 28*, 4 (2009), 1247–1155.

[DBB03] DUTRE P., BEKAERT P., BALA K.: *Advanced Global Illumination*. A K Peters, 2003.

[EMP*03] EBERT D., MUSGRAVE K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*. Morgan Kauffmann, 2003.

[Fat09] FATTAL R.: Participating media illumination using light propagation maps. *ACM Trans. Graph. 28*, 1 (2009), 1–11.

[FTK86] FUJIMOTO A., TAKAYUKI T., KANSEI I.: Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications 6*, 4 (1986), 16–26.

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. *SIGGRAPH '98 Proceedings* (1998), 311–320.

[LBC95] LANGUENOU E., BOUATOUCH K., CHELLE M.: Global illumination in presence of participating media with general properties. In *Eurographics Workshop on Rendering* (1995), pp. 69–85.

[Lep10] LEPPANENA J.: Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code. *Annals of Nuclear Energy* (2010). In Press.

[Lew89] LEWIS J. P.: Algorithms for solid noise synthesis. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (1989), pp. 263–270.

[Per85] PERLIN K.: An image synthetisizer. In *Computer Graphics (SIGGRAPH '85 Proceedings)* (1985), pp. 287–296.

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

[QXFN07] QIU F., XU F., FAN Z., NEOPHYTOS N.: Lattice-based volumetric global illumination. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1576–1583.

[RSK08] RAAB M., SEIBERT D., KELLER A.: Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 2008, pp. 591–606.

[RT87] RUSHMEIER H., TORRANCE K.: The zonal method for calculating light intensities in the presence of a participating medium. In *SIGGRAPH 87* (1987), pp. 293–302.

[Rus94] RUSHMEIER H.: Rendering Participating Media: Problems and Solutions from Application Areas. In *Proceedings of the 5th Eurographics Workshop on Rendering* (1994), pp. 35–56.

[SKTMC10] SZIRMAY-KALOS L., TÓTH B., MAGDICS M., CSÉBFALVI B.: Efficient free path sampling in inhomogeneous media. In *Eurographics Poster* (2010).

[WMHL65] WOODCOCK E., MURPHY T., HEMMINGS P., LONGWORTH S.: Techniques used in the GEM code for Monte Carlo neutronics calculation. In *Proc. Conf. Applications of Computing Methods to Reactors, ANL-7050* (1965).