Line Space Gathering for Single Scattering in Large Scenes

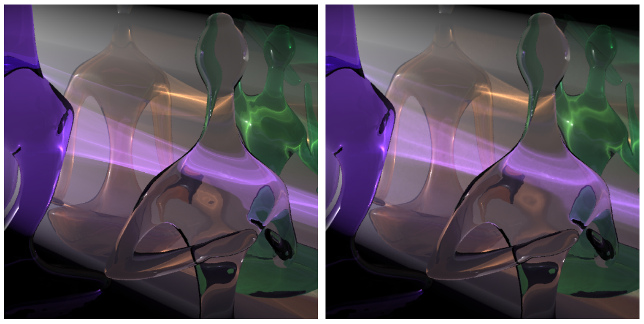
Xin Sun* Kun Zhou† Stephen Lin* Baining Guo*

*Microsoft Research Asia †State Key Lab of CAD&CG, Zhejiang University

Abstract

We present an efficient technique to render single scattering in large scenes with reflective and refractive objects and homogeneous participating media. Efficiency is obtained by evaluating the final radiance along a viewing ray directly from the lighting rays passing near to it, and by rapidly identifying such lighting rays in the scene. To facilitate the search for nearby lighting rays, we convert lighting rays and viewing rays into 6D points and planes according to their Plücker coordinates and coefficients, respectively. In this 6D line space, the problem of closest lines search becomes one of closest points to a plane query, which we significantly accelerate using a spatial hierarchy of the 6D points. This approach to lighting ray gathering supports complex light paths with multiple reflections and refractions, and avoids the use of a volume representation, which is expensive for large-scale scenes. This method also utilizes far fewer lighting rays than the number of photons needed in traditional volumetric photon mapping, and does not discretize viewing rays into numerous steps for ray marching. With this approach, results similar to volumetric photon mapping are obtained efficiently in terms of both storage and computation.

Keywords: single scattering, ray tracing, spatial hierarchy, Plücker coordinates and coefficients



(a) LSG
26 min (CPU), 3.7 min (GPU)
404 MB (CPU), <800 MB(GPU)

(b) VPM
729 min (CPU)
7.1 GB (CPU)

Figure 1: *Statuettes, rendered by line space gathering (LSG) and volumetric photon mapping (VPM). The scene contains* $41K$ *triangles and is illuminated by a single point light source. The full images have a resolution of* 512×512 *with* 2×2 *supersampling. CPU and GPU performance is measured on a Dell PowerEdge 2900 with dual Intel Xeon X5470 3.33 GHz quad-core CPUs and an NVIDIA GeForce GTX 280 graphics card, respectively.*

1 Introduction

Single scattering in participating media becomes significantly more challenging to render when occlusions and specular bounces are involved. There exist two principal reasons for this. The first is the increased complexity of light paths when these additional forms of light interaction are present. This produces various visual effects such as glows around light sources [Sun et al. 2005], volumetric shadows and light shafts [Max 1986], and volumetric caustics. The second reason is that the visual effects of single scattering are typically high frequency [Walter et al. 2009], and thus require dense sampling of light flux in order to render them correctly. This dense sampling leads to substantial storage and computation costs, limiting both performance and the size of scenes that can be rendered in practice. In our work, we propose an algorithm that is efficient in rendering the high frequency effects of single scattering in large scenes of homogeneous participating media, while accounting for light interactions such as occlusions and multiple specular bounces.

In prior work, the only effective approach for processing complex light paths while maintaining high rendering quality is volumetric photon mapping [Jensen and Christensen 1998]. However, a scene full of fine details and high frequency lighting effects leads to a prohibitive cost in computation and memory. Our method can reduce the heavy load of computation and memory by more than one order of magnitude. This reduction in memory consumption is important because it allows for an implementation on GPUs, which brings another order of magnitude improvement in performance. For example, rendering the scene of Figure 1 by volumetric photon mapping takes more than 7 GBytes of memory and 12 hours of computation time on a PC with dual Intel Xeon X5470 3.33 GHz quad-core CPUs, but takes only 3.7 minutes on an NVIDIA GeForce GTX 280 graphics card with our algorithm. This acceleration is gained while maintaining comparable rendering quality.

The key ingredient of our algorithm is line space gathering, a novel and efficient method for collecting single scattering flux along viewing rays. This procedure is performed in a parametric line space containing the lighting and viewing rays. Rather than trace numerous photons and search for them at sample points along viewing rays, our method identifies points at which lighting rays and viewing rays intersect or nearly intersect, and evaluates radiance transport directly at these points where the scattering events occur. Since the number of lighting and viewing rays is considerably smaller than the number of photons and sample points in volumetric photon mapping, this method offers much greater efficiency in terms of both storage and computation. Typically, the number of lighting rays is about 1% of the number of photons, with rendering results of similar quality.

The main challenge of line space gathering lies in rapidly finding the points where lighting and viewing rays approximately intersect. Although it is straightforward to determine whether two given lines intersect, how to efficiently query for lines (lighting rays) that nearly intersect a given line (viewing ray) is an open problem, and one that increases in complexity when considering rays that undergo many direction changes within the scene. A search for nearest line segments is fundamentally different from that for nearest points used in volumetric photon mapping. This is because an effective hierarchy of the 3D space cannot in general be constructed for line segments due to their length in the scene, which leads to many intersections with splitting planes.

We address this problem by employing a parametric line space to represent ray segments, and constructing a spatial hierarchy within this space. Ray segments are approximated by the infinite lines they lie on, and are converted to the 6D parametric space of Plücker coordinates and coefficients in which the lighting rays and viewing rays are represented as points and planes, respectively. In this 6D parametric space of rays, a spatial hierarchy can be effectively constructed on the 6D points of lighting rays, and the search becomes a plane query within a set of points. Since this search finds intersections between lines instead of line segments, it is used to efficiently cull lighting ray segments for a given viewing ray segment, leaving a relatively small number of segments for a more computationally intensive direct intersection evaluation. At the intersection points, the radiance transport can be rapidly evaluated from the initial flux at the beginning of each segment, since the attenuation and scattering coefficients are constant in a homogeneous medium.

Our technique of line space gathering is not limited to certain types of light paths or volumes of fixed resolution. Also, it can be easily integrated into the ray tracing framework as a substitute for volumetric photon mapping to render single scattering in a homogeneous volumetric material. In addition, as exemplified in Figure 8, this work can be combined with other techniques such as photon mapping and shadow maps to efficiently handle single scattering within a more comprehensive rendering system.

2 Related works

Monte-Carlo ray tracing and volumetric photon mapping Monte Carlo ray tracing [Kajiya 1986] offers the most general solution for rendering various types of light interactions, but the high computational complexity precludes its use for radiance computation in volumes. Based on bidirectional Monte-Carlo ray tracing, volumetric photon mapping [Jensen and Christensen 1998] provides an efficient, physically accurate solution for participating media using photons to represent flux. However, the storage of numerous photons and the cost of radiance computation from them limit performance and the scale of rendered scenes.

Optimizations of these techniques have been presented for faster rendering. In [Jarosz et al. 2008b], point queries are replaced with a beam query in the photon gathering pass, to avoid discretization of straight viewing rays. Our method also employs a beam query, but instead gathers the flux contributions of lighting rays. In [Jarosz et al. 2008a], radiance caching is shown to bring significant speedups in Monte Carlo ray tracing of large scenes with participating media. Light paths containing multiple specular bounces, however, cannot be processed in this way.

Gathering points for photon mapping may be sampled at non-regular intervals along a viewing ray. In the hair rendering method of [Moon and Marschner 2006], the gathering points lie only on the hair fibers. For our technique, flux is gathered only at points where the viewing ray is in close proximity to a lighting ray.

Eikonal rendering with volume representation A common alternative approach to interactive or real-time performance has been to employ approximations on the scene. One such approximation is to discretize the scene as a fixed resolution volume, with scattering parameters and the flux of lighting rays stored in voxels. Flux transport among the voxels is then computed according to a first order approximation of the Eikonal equation, as well as reflection and refraction [Ihrke et al. 2007; Sun et al. 2008]. In comparison to processing an enormous number of photons, voxel based computation significantly reduces rendering and storage costs. But since rendering performance depends greatly on the resolution of the volume, large scenes and complex geometry cannot be processed in practice without a significant loss of rendering quality.

Explicit solution of light paths A second approximation is to consider only certain types of scenes in which light paths can be explicitly solved [Mitchell and Hanrahan 1992; Chen and Arvo 2000; Iwasaki et al. 2002; Ernst et al. 2005; Krüger et al. 2006; Hu et al. 2010; Walter et al. 2009]. For example, the work of [Walter et al. 2009] renders transparent objects whose convex hull is bounded by a triangle mesh. Scenes for which explicit solution is possible generally exclude many visual effects, such as light source glows and volumetric caustics seen through reflectors and refractors.

Real-time single scattering for fog, volumetric shadows and volumetric caustics Methods for real-time rendering of fog efficiently account for single scattering but assume straight, unoccluded rays [Shreiner et al. 2005; Sun et al. 2005; Zhou et al. 2007]. To rapidly render high-quality volumetric shadows, adaptive sampling of viewing rays [Wyman and Ramsey 2008] and media volumes [Ren et al. 2008] has been employed. In [Zhou et al. 2008b], volumetric shadows are generated with environment lighting.

Volumetric caustics present a greater challenge because light paths often include multiple specular bounces such as reflection and refraction. Some techniques assume lighting rays to undergo only a limited number of reflections or refractions [Iwasaki et al. 2002; Ernst et al. 2005], and cannot handle optically complex scenes such as Figure 7 where many rays have 15 bounces or more. To handle complex light paths, lighting rays can be refracted multiple times in camera space, and then splatted directly to screen pixels [Krüger et al. 2006]. Attenuation of the refracted lighting rays can be also incorporated [Hu et al. 2010; Papadopoulos and Papaioannou 2009]. Many GPU aided methods of this kind employ rasterization to accelerate ray marching [Ernst et al. 2005; Krüger et al. 2006; Hu et al. 2010; Papadopoulos and Papaioannou 2009], but must assume no specular interactions of viewing rays so that their positions in the perspective space can be determined.

Parametric line spaces Line primitives have been used in various graphics problems besides volumetric caustics rendering. In the visibility tests of [Teller 1992; Bittner 2003], Plücker coordinates are used to determine whether a line lies within a given spatial domain, i.e., evaluating *membership* to a set of lines. By contrast, our method seeks to determine *nearness* of a line, since in practice the probability of exact intersections between lighting and viewing rays approaches zero. To evaluate nearness of lines in the primal space, we derive a corresponding measure in Plücker coordinates that can be efficiently computed. Light rays nearest to a surface point are gathered in [Havran et al. 2005] for computing global illumination. Since point queries are used, gathering is efficiently performed directly in the primal space, rather than in a parametric line space.

3 Radiance Estimation of Single Scattering

Within a participating medium, the radiance transfer between two points from x_0 to x along a line segment with direction $\overrightarrow{\omega}$ can be expressed as

$$L(x, \overrightarrow{\omega}) = \tau(x_0, x)\, L(x_0, \overrightarrow{\omega}) + L_s(x, x_0) \qquad (1)$$

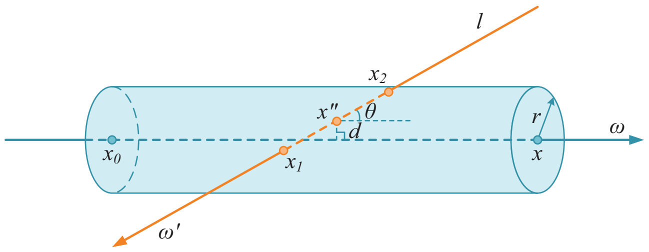


Figure 2: *For single scattering, radiance transfer is evaluated at the intersection points between lighting rays and viewing rays.*

$$L_s\left(x, x_0\right) = \int_{x_0}^{x} \tau\left(x', x\right) \sigma\left(x'\right) L_i\left(x', \overrightarrow{\omega}\right) dx' \quad (2)$$

$$L_i\left(x', \overrightarrow{\omega}\right) = \int_{\Omega} f\left(x', \overrightarrow{\omega}', \overrightarrow{\omega}\right) L\left(x', \overrightarrow{\omega}'\right) d\omega'. \quad (3)$$

Following the notation in [Jensen and Christensen 1998], $L\left(x, \overrightarrow{\omega}\right)$ represents the radiance arriving at point x along direction $\overrightarrow{\omega}$. $L_i\left(x', \overrightarrow{\omega}\right)$ is the in-scattered radiance at point x' towards x. $\tau\left(x', x\right)$ is the transmittance between these two points, computed as $e^{-\int_{x'}^{x} \kappa(\xi) d\xi}$. κ denotes the extinction coefficient, which is the sum of the scattering coefficient σ and the absorption coefficient α. $f\left(x', \overrightarrow{\omega}', \overrightarrow{\omega}\right)$ is the normalized phase function that determines the relative proportion of light arriving at x' from direction $\overrightarrow{\omega}'$ that is scattered toward direction $\overrightarrow{\omega}$.

Traditional volumetric photon mapping [Jensen and Christensen 1998] uniformly samples the values of L_i in Equation (3) along the viewing rays. At each sample point, L_i is computed by gathering the energy of all photons within a radius r:

$$L_i\left(x, \overrightarrow{\omega}\right) \approx \frac{1}{\sigma\left(x\right)} \sum_{p=1}^{n} f\left(x, \overrightarrow{\omega_p}', \overrightarrow{\omega}\right) \frac{\Delta\Phi_p\left(x, \overrightarrow{\omega_p}'\right)}{\frac{4}{3}\pi r^3} \quad (4)$$

where the $\Delta\Phi_p\left(x, \overrightarrow{\omega_p}'\right)$ is the flux carried by a photon p at x with a direction $\overrightarrow{\omega_p}'$.

In volumetric photon mapping with beam query [Jarosz et al. 2008b], Equation (2) is directly evaluated for a line segment, instead of solving Equation (3) with multiple samples as in Equation (4). The energy of all photons is gathered within a cylinder centered about the line segment between x and x_0 with a fixed radius r:

$$L_s\left(x, x_0\right) \approx \sum_{p=1}^{n} \tau\left(x', x\right) f\left(x', \overrightarrow{\omega_p}', \overrightarrow{\omega}\right) \frac{\Delta\Phi_p\left(x, \overrightarrow{\omega_p}'\right)}{\pi r^2 \left\|x - x_0\right\|}. \quad (5)$$

As illustrated in Figure 2, the radiance transfer for single scattering should be evaluated at the intersection points between lighting rays and viewing rays. At an intersection, some flux from the lighting ray is lost along the viewing ray during scene traversal. Analogous to the beam gathering of photon energies in Equation (5), the radiance contributed by a lighting ray to the viewing ray as shown in Figure 2 can be computed as

$$L_r\left(x, x_0, l\right) = \int_{x_1}^{x_2} w\left(x', x, \overrightarrow{\omega}', \overrightarrow{\omega}\right) \Phi_l\left(x', \overrightarrow{\omega}'\right) dx' \quad (6)$$

$$w\left(x', x, \overrightarrow{\omega}', \overrightarrow{\omega}\right) = \frac{\tau\left(x', x\right) \sigma\left(x'\right) f\left(x', \overrightarrow{\omega}', \overrightarrow{\omega}\right)}{\pi r^2 \left\|x - x_0\right\|} \quad (7)$$

where is $\overrightarrow{\omega}'$ is the direction of the lighting ray l, and $\Phi_l\left(x', \overrightarrow{\omega}'\right)$ is the flux it carries at x' along the ray direction. Since the radius of the beam is very small, we consider the flux of the ray and the traversed volumetric material to be approximately constant, such that $w\left(x', x, \overrightarrow{\omega}', \overrightarrow{\omega}\right)$ can be moved outside of the integral:

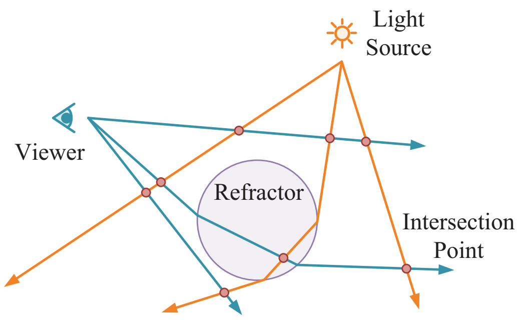


Figure 3: *Rendering with nearest line search of line queries. We first trace rays from the light sources and viewpoint. Then we gather radiance from the lighting rays at the intersection points for each viewing ray. The radiances along viewing rays are deposited to corresponding pixels to generate the final image.*

$$L_r\left(x, x_0, l\right) \approx w\left(x'', x, \overrightarrow{\omega}', \overrightarrow{\omega}\right) \Phi_l\left(x'', \overrightarrow{\omega}'\right) \frac{\left(r^2 - d^2\right)^{\frac{1}{2}}}{sin\theta} \quad (8)$$

where x'' is the point on the lighting ray l closest to the line segment between x and x_0, and d is the corresponding distance. The angle between the two rays is denoted by θ, which is clamped to a minimum threshold to avoid division by zero. From Equation (8), we obtain a simple approximation of Equation (2) as the summed radiance contributed by all lighting rays passing through the cylinder:

$$L_s\left(x, x_0\right) \approx \sum_{l=1}^{n} L_r\left(x, x_0, l\right). \quad (9)$$

In this formulation, it is assumed that the closest points between the lighting and viewing rays do not lie at endpoints of a ray segment. Since ray segments in a scene are generally much longer than the gathering radius r, such cases have little effect on the final solution and can be safely ignored.

Although in theory the radiance computation should consider only exact intersections between lighting rays and viewing rays, a kernel of radius r is needed in practice since the probability of two lines intersecting each other approaches zero. Equations (1) and (9) allow for direct computation of radiance transport between lighting rays and viewing rays without an intermediate representation of photons and sampling points, as shown in Figure 3. We capitalize on the efficiency obtained using line-to-line intersections in the following steps of our proposed algorithm:

1. Construct the spatial hierarchy on the geometry of the scene.
2. Trace rays from light sources and viewpoint. The rays may be reflected and refracted as they traverse the scene. Compute the initial flux of each lighting ray segment and the initial transmittance-to-viewpoint of each viewing ray segment.
3. Convert the lighting and viewing rays to the parametric line space, and construct a spatial hierarchy on the lighting rays.
4. In the parametric line space, gather the lighting rays that intersect each viewing ray and compute the radiance transfer between them. The final radiances of the viewing rays are deposited to the corresponding shading pixels to produce the final image.

In our work, we employ a kd-tree [Zhou et al. 2008a; Hou et al. 2009] for the first step. For the second step, the rays from light sources are generated in a manner similar to photon mapping, with initial flux set to a constant, and density and directions determined by the energy emission distribution of the light sources. The last two steps deal with how to efficiently search for the nearest lighting rays for each viewing ray, and are described in detail in the following section.

4 Spatial Hierarchy in Parametric Line Space

For our algorithm, high performance necessitates efficient search for nearest line segments between lighting and viewing rays. Most problems of nearest neighbor search in low dimensions can be accelerated by spatial hierarchies. However, the line segments of rays generally cross large portions of the scene and are too long to be efficiently clustered by space splitting techniques such as kd-trees. Furthermore, the orientations of line segments are often far from those of axis aligned planes, leading to primitives with large bounding boxes in a bounding volume hierarchy (BVH).

Spatial hierarchies generally favor small primitives, ideally a set of points. In this work, we utilize Plücker coordinates to convert lighting rays to points, and construct spatial hierarchies on points instead of line segments. Our solution is inspired by a recent study on nearest affine subspace search in the area of pattern recognition [Basri et al. 2009]. Their method maps subspaces to points in a higher dimensional space prior to nearest neighbor search. It is, however, unsuitable for our application for three reasons. First, the distance between two affine subspaces is fundamentally different from our distance of two straight lines. Second, the data set is converted to points of very high dimensions, 11D or more, in which processing incurs high memory and computation costs. Third, a large constant offset is introduced in the distance computation, which slows traversal through the hierarchy and reduces floating-point precision. We instead develop a nearest line search tailored to our work.

4.1 Review of Plücker Coordinates and Coefficients

Introduced by Julius Plücker in the 19th century, Plücker coordinates and coefficients are widely used in computational geometry [Stolfi 1988]. Given a oriented straight line l in 3D space defined by two points $a = (a_1, a_2, a_3, a_4)$ and $b = (b_1, b_2, b_3, b_4)$ in homogeneous coordinates, we can compute the corresponding Plücker coordinates $\pi(l)$ and coefficients $\varpi(l)$ as

$$\pi(l) = \left(l_{\{1,2\}}, l_{\{1,3\}}, l_{\{1,4\}}, l_{\{2,3\}}, l_{\{2,4\}}, l_{\{3,4\}}\right) \quad (10)$$

$$\varpi(l) = \left(l_{\{3,4\}}, -l_{\{2,4\}}, l_{\{2,3\}}, l_{\{1,4\}}, -l_{\{1,3\}}, l_{\{1,2\}}\right) \quad (11)$$

$$l_{\{i,j\}} = a_i b_j - a_j b_i. \quad (12)$$

Because of geometric duality, each straight line in 3D space maps to a point in 6D space as its Plücker coordinates and a hyperplane passing through the origin in 6D space with its Plücker coefficients.

The incidence of lines in 3D space can be easily determined from their Plücker coordinates and coefficients. Consider two lines l and l' determined by four vertices a, b and a', b' as shown in Figure 4. The dot product of $\pi(l)$ and $\varpi(l')$ can be expressed as the following 4×4 determinant of the four vertices:

$$\pi(l) \bullet \varpi(l') = \begin{vmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ a_1{}' & a_2{}' & a_3{}' & a_4{}' \\ b_1{}' & b_2{}' & b_3{}' & b_4{}' \end{vmatrix}. \quad (13)$$

If the two lines lie on the same plane, this determinant is equal to zero. Geometrically, this is because the determinant has a rank of at most three if one of the vertices lies on the plane determined by the other three vertices. Except for the special case of parallel lines, a zero determinant indicates an intersection between the two lines.

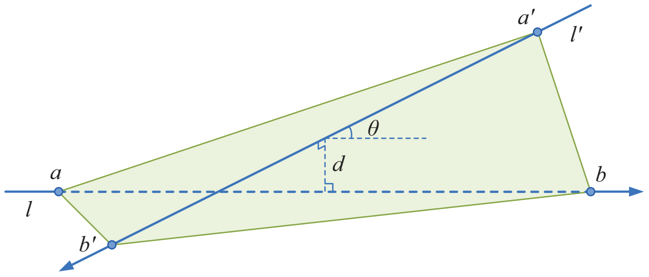


Figure 4: *Two 3D lines determined by four vertices. The volume of the tetrahedron constructed by the four vertices is used as our measure of distance between the two straight lines.*

4.2 Distance Computation between Straight Lines

Instead of searching for straight lines that exactly intersect, our method seeks lines nearest to a viewing ray. The distance between lines is not equal to the determinant of Equation (13), so we derive an approximation that is related to it.

We first convert the four vertices that define the two lines, shown in Figure 4, from homogeneous coordinates to their equivalent Cartesian coordinates with the last coordinate value set to one. With this conversion, the absolute magnitude of the determinant in Equation (13) becomes equal to one-sixth of the volume of the tetrahedron defined by the four vertices.

The volume of the tetrahedron is not solely determined by the distance between the two straight lines, but is related as follows:

$$\begin{aligned} V_{\{a,b,a',b'\}} &= \frac{1}{6}\left|\pi(l) \bullet \varpi(l')\right| \\ &\geq \frac{1}{6} \| \ a - b \ \| \| \ a' - b' \ \| \, d \, sin\theta \end{aligned} \quad (14)$$

where d and θ are the distance and the angle between the two straight lines. This inequality becomes an equality when the closest points of the two lines lie on the line segments (a, b) and (a', b').

For each of the two lines, we position the vertices at a fixed distance c apart such that (a, b) and (a', b') span the scene. If the closest points on the two lines exist within the scene, the lower bound of d in Equation (14) becomes

$$d \geq \frac{6\, V_{\{a,b,a',b'\}}}{c^2} = \frac{|\pi(l) \bullet \varpi(l')|}{c^2} \quad (15)$$

which is used as our distance metric between two straight lines. Since this measure provides only a lower bound on distance, we use it to gather all the lines that potentially lie near to a viewing ray.

After gathering lines in this manner, we explicitly compute the distance between the line segments to remove cases where the actual distance is greater than r or the closest points lie outside the viewed scene. With this approach, considerable computation is saved by performing the exact distance calculation on only a small subset of all the line segment pairs.

4.3 Spatial Hierarchy Construction

Our distance measure, the scaled absolute magnitude of the determinant in Equation (13), represents the distance in 6D space

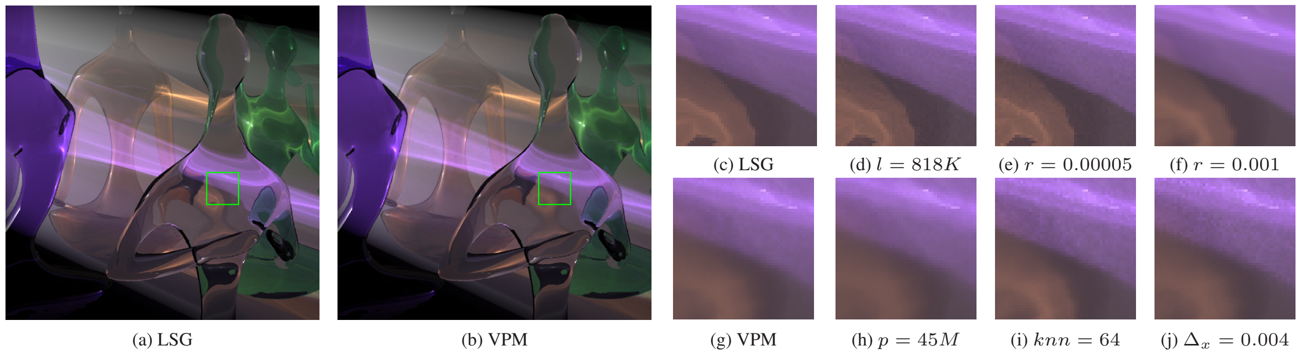


Figure 5: *Comparison of different parameters settings for line space gathering (LSG) and volumetric photon mapping (VPM). (a)(b) are the same as Figure 1(a)(b). (c)(g) are closeups of the green boxes in (a)(b), respectively, and have parameter settings of* $l = 1,636K$*,* $r = 0.0001$*;* $p = 89M$*,* $knn = 128$*,* $\Delta_x = 0.002$*. (d) to (f) are results of LSG with different parameter values, as are (h) to (j) for VPM.*

of a point with coordinates $\pi(l)$ and a hyperplane with coefficients $\varpi(l')$. We convert the lighting rays to 6D points by their Plücker coordinates, and the viewing rays to 6D hyperplanes by their Plücker coefficients. Obtaining the nearest lighting rays to a given viewing ray is then formulated as a problem of finding the nearest points to the corresponding hyperplane in the 6D space.

To efficiently identify the nearest points, we construct a hierarchy of a perfect 8-ary tree in the 6D space. After converting the lines to corresponding Plücker coordinates, we use two steps to construct the hierarchy:

1. We construct a balanced kd-tree with median splitting. Each leaf node contains at least 4 points.
2. We collapse every three non-leaf levels into one level. Thus each non-leaf node has exactly 8 children.

The constructed tree is stored in the breadth-first search order. Each node stores its bounding box as 12 floating-point values. According to the order of the corresponding leaf nodes in the constructed tree, the 6D points are sorted and deposited consecutively in a buffer. Each leaf node contains two additional integers to represent its range of indices in the buffer. We traverse the hierarchy in depth-first order. The buffer locations of parents, children and siblings can be determined from a node's position since the hierarchy is a perfect 8-ary tree. This technique is a form of stackless traversal. In the GPU implementation, each traversal thread keeps only the current node and its index in registers or shared memory; local memory is not required.

The hierarchy culls about 99.5% of line intersection tests compared with exhaustive testing. Among the line intersections that remain after culling, about 10% of them are valid. Note that we tried many different spatial hierarchy candidates before reaching the current solution. Construction in primal space failed as expected. In line space, we have tried octree, uniform grid, traditional BVH and kd-tree with heuristics of surface area (SAH), voxel volume (VVH) and bounding aspect ratio (BAR). We have also tried different numbers of children for non-leaf nodes, and storing the splitting dimension and splitting position instead of the bounding box. These different candidates may use less storage or cull more intersection tests, but all have performance much worse than our current solution. The other approaches are unsuitable for our work mainly because we use plane queries in line space, rather than point queries, which consume significant time on inner node traversal. An exception is BAR, but its complex data structure is inefficient for GPU processing. By contrast, the perfect 8-ary trees that we use allow for regular memory accesses.

5 Experiments and Results

We implemented a GPU version of the described algorithms in NVidia's CUDA on an NVIDIA GeForce GTX 280 graphics card with 1 GByte of memory. We also implemented a CPU version on dual Intel Xeon X5470 3.33 GHz quad-core CPUs for performance comparisons to volumetric photon mapping, which was also executed on this platform. All images shown in this paper have a resolution of 512×512 with 2×2 supersampling, except for Figure 8 which has 4×4 supersampling.

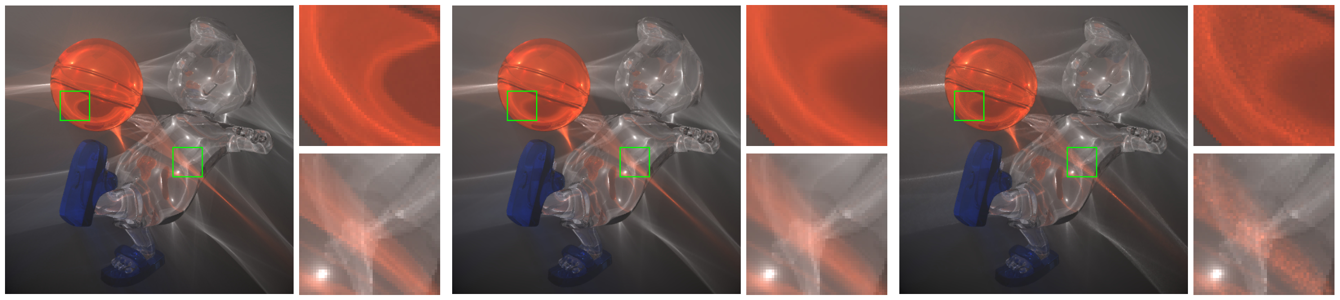
Renderings with our line space gathering (LSG) technique are compared to volumetric photon mapping (VPM) in Figures 1, 6 and 7. We denote the parameters for LSG as l for the number of lighting rays and r for the gathering radius. The parameters for VPM are p for the number of photons, knn for the number of photons gathered at each sampling point, and Δ_x for the ray marching step size, with the maximum bounding box dimension of the scenes set to 1.

Complex lighting features are generated among the four statuettes in Figure 1. Here, we disregard reflections of lighting rays in order to emphasize the effects of refraction, such as the high frequency volumetric caustics seen directly through the refractors. LSG in (a) produces results comparable to VPM in (e) with a speedup of more than two orders of magnitude.

We examine how changes in parameter settings affect our LSG algorithm and VPM in Figure 5, the scene of four statuettes exhibited in Figure 1. At the original parameter settings, LSG in Figure 5(a) produces results comparable to VPM in (b) with a speedup of more than two orders of magnitude. Closeups of (a) and (b) are shown in (c) and (g), respectively.

The parameters l in LSG and p in VPM each control the number of light flux samples in their respective algorithms. Reducing these values elevates noise in (d), while increasing blur in (h). This difference is due to the fixed value of knn, which leads to a larger sampling radius when the number of photons is reduced. The parameters r in LSG and knn in VPM are also related in that they both affect the breadth of flux sampling in radiance computation. Decreasing these two parameters in (e) and (i) increases sharpness and noise in both cases. However, the structure of noise differs according to flux representation. While VPM exhibits grainy effects, subtle line patterns appear in LSG. When r is increased in (f), there exists more blurring in LSG.

Usually the gathering radius r is quite dependent on the number of lighting rays l. With more lighting rays launched, a smaller radius is used. If a small number of rays and a large radius are used, the

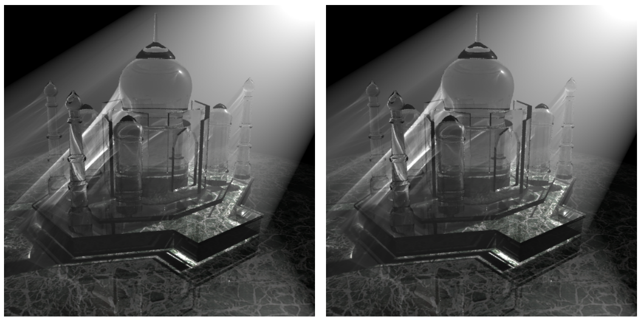


(a) LSG
73 min, 354 MB (CPU); 6.5 min (GPU)

(b) VPM
438 min, 7.1 GB (CPU)

(c) VPM
7.3 min, 1.8 GB (CPU)

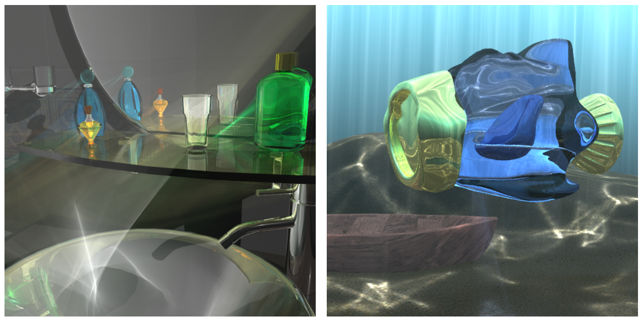
Figure 6: *Soccer player. The scene contains $43K$ triangles and is illuminated by three point light sources. Coefficients of the participating media are $\kappa = 0.3$, $\sigma = 0.2$. Parameter settings are (a) $l = 1,535K$, $r = 0.0005$; (b): $p = 92M$, $knn = 128$, $\Delta_x = 0.002$; (c): $p = 24M$, $knn = 16$, $\Delta_x = 0.01$.*



(a) LSG
53 min, 229 MB (CPU); 5.3 min (GPU)

(b) VPM
843 min, 4.5 GB (CPU)

Figure 7: *Crystal Taj Mahal. The scene contains $53K$ triangles and is illuminated by a point light source. Coefficients of the participating medium are $\kappa = 0.3$, $\sigma = 0.2$. Parameter settings are (a) $l = 677K$, $r = 0.0003$; (b) $p = 64M$, $knn = 128$, $\Delta_x = 0.001$.*



(a) Washroom. 31.7 min (GPU)

(a) Underwater. 30.5 min (GPU)

Figure 8: *Two complex scenes rendered with a combination of techniques including shadow mapping, photon mapping and our line space gathering. Various lighting effects, such as shadows and caustics, are generated both on surfaces and in volumes. Coefficients of the participating media in both scenes are $\kappa = 0.03$ and $\sigma = 0.02$. Scale of the scenes: (a) $72K$ triangles; (b) $93K$ triangles. Parameters of line space gathering: (a) $l = 2.6M$, $r = 0.0001$; (b) $l = 8.9M$, $r = 0.0005$.*

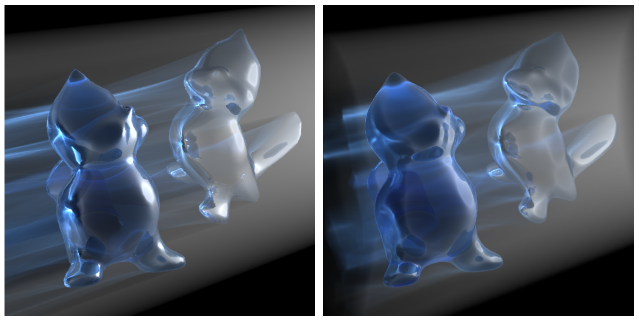
final image becomes quite blurry. These parameters are difficult to set based on physical properties. In our experiments, the radius is set to be large enough to avoid artifacts from under-sampling. If the result is blurry, we launch more rays and reduce the gathering radius. This scheme is similar to setting the photon number p and knn in VPM.

Both reflection and refraction are employed in rendering the soccer player of Figure 6, producing a multitude of high frequency lighting effects. Volumetric caustics are visible from reflectors as well as refractors, such as the red caustics generated by the soccer ball that are reflected and refracted by the player's body. For viewing rays that reflect towards volumetric caustics, bright projections of the caustics are found at the reflection points, such as the bright spot on the left side of the ball. Besides volumetric caustics, glows around light sources are reflected on the objects as well. For the given parameter settings, LSG and VPM yield similar renderings as shown in (a) and (b). But when the VPM parameters are adjusted to match the performance of LSG, there is a large degradation in quality, exhibited in (c).

Line space gathering can be easily combined with other popular ray tracing techniques, as demonstrated for a crystal Taj Mahal in Figure 7. In (a), LSG is used to render volumetric caustics while photon mapping is applied to render surface caustics. Since the crystal Taj Mahal is composed of many components, light travels through the scene along complex paths with many specular interactions. The intricacy of flux transport in this example leads to various fine-scale details. Even though many of the lighting rays consist of numerous short line segments, this method nevertheless is considerably more efficient in computation and storage than volumetric photon mapping shown in (b).

Two additional examples that apply line space gathering in conjunction with other rendering techniques are exhibited in Figure 8. In the two scenes, single scattering in homogeneous media is rendered by our method, while the volumetric shadows and light shafts are computed by shadow mapping, and the surface caustics by photon mapping. In (a), the fog effects result from both direct lighting from a window and reflected lighting by the mirror and metallic basin. The cyan bottle generates volumetric caustics in two directions corresponding to the direct lighting and mirror reflected lighting. In (b), the reflections and refractions of the submerged submarine generate caustics generated both in the water and on the sand. The top part of the submarine exhibits reflections of volume caustics, while the bottom of the submarine reflects caustics from the surface below. The scene is illuminated by indirect and volumetric ambient lighting.

We also compare the differences in rendering quality between line space gathering and Eikonal rendering with adaptive photon tracing



(a) LSG, 44.8 sec(GPU) (b) Eikonal rendering, 200 ms(GPU)

Figure 9: *Comparison of line space gathering and Eikonal rendering with adaptive photon tracing. Coefficients of the participating medium are $\kappa = 0.02$, $\sigma = 0.01$. Scale of the scenes: (a) 10K triangles; (b) the volume resolution is 128 × 128 × 128. Parameter settings are (a) $l = 1M$, $r = 0.001$; (b) $1M$ photons are shot with about half of them hitting a refractive object, and the marching step size in the light and view passes is a half voxel.*

[Sun et al. 2008], in Figure 9. To maximize the quality of the object models, the volume for Eikonal rendering is arranged to tightly bound the two Tweety models. We note that with this scene and parameter settings, increasing the photon count and decreasing the step size does not improve Eikonal rendering. Several differences in the rendering results of the two algorithms are evident. In the volumetric caustics, LSG is seen to be appreciably sharper than Eikonal rendering, which is limited by the volume resolution. The bounds of the Eikonal rendering volume also contribute to lower rendering quality. At the left side of (b), the volume caustics are clipped because they extend outside of the rendering volume. The volume bounds also degrade results in an indirect manner, since media scattering that occurs outside of the volume does not provide radiance to the scene, as exemplified by the weak surface reflections on the right sides of the Tweety models. By contrast, the line representation used in LSG does not have these issues.

Performance and memory comparisons between our technique and volumetric photon mapping are listed in Table 1. For CPU implementations of both algorithms with comparable levels of rendering quality, our line space gathering uses more than an order of magnitude less memory, and also has about one order of magnitude higher rendering performance. With our GPU implementation of LSG, performance improves by another order of magnitude. To utilize the parallelism of GPUs, a large number of viewing rays should be produced before line space gathering, but this requires considerable temporary storage. We employ a memory bounding scheme as in [Hou et al. 2009] to obtain a good tradeoff between parallelism and the scale of scenes. We set the bound on GPU memory usage to 800 MBytes so that the algorithm can run on an NVIDIA GeForce GTX 280 graphics card with 1 GByte of memory. For the scene in Figure 8 (b), there are too many lighting rays to fit in video memory, so we divide them into four parts and perform gathering on them one by one. Because we use a fixed gather radius of r, the sum of the four results is equivalent to processing all the lighting rays together.

We present comparisons of our method to [Jensen and Christensen 1998] instead of [Jarosz et al. 2008b] mainly for two reasons. First, [Jensen and Christensen 1998] is widely used as a ground truth for rendering quality, so we compare our renderings with it directly. Second, we focus on high quality rendering of large scenes, which requires a considerable number of photons, and the memory needed for [Jarosz et al. 2008b] is more than double that of [Jensen and Christensen 1998]. For most of our scenes, such as the one of statuettes in Figure 1, [Jarosz et al. 2008b] requires more than 16GBytes of memory, which is more than our workstation and most popular desktops/workstations (2-4GBytes RAM) can handle. Note that based on reported statistics, [Jarosz et al. 2008b] brings a 5×-50× improvement in performance, while our GPU method yields a 70×-200× improvement. This efficient GPU implementation can be attributed to the much lower memory consumption of our method.

Figure	Line Space Gathering			Volumetric Photon Mapping	
	Memory MBytes (CPU)	Time Minutes (CPU)	Time Minutes (GPU)	Memory GBytes (CPU)	Time Minutes (CPU)
1	404	26	3.7	7.1	729
6	354	73	6.5	7.1	438
7	229	53	5.3	4.5	843
8 (a)	NA	NA	31.7	NA	NA
8 (b)	NA	NA	30.5	NA	NA
9 (a)	NA	NA	0.75	NA	NA

Table 1: *Performance and memory comparisons between our technique of line space gathering and volumetric photon mapping.*

In our experiments, the ray shooting and hierarchy construction of LSG takes less than 5 seconds on GPUs and less than 1 minute on CPUs. The photon shooting and kd-tree construction of VPM takes about 10 minutes on CPUs. For LSG, generally most of the time consumption is for gathering. Since the pre-process is just a small part of the run-time computation, it was not carefully optimized. We aimed only to improve the quality of the spatial hierarchy without taking the construction cost into account. The reported performance does not include the preprocessing time because the pre-process takes substantial time for VPM on the CPU in the case of Figure 6 (c), where performance is set to match that of LSG in Figure 6 (a). In fact, the preprocessing time for VPM in this case actually exceeds the rendering time, making it hard for an "equal performance" comparison to be made.

6 Conclusion

We have presented a technique for efficient rendering of single scattering in large scenes with homogeneous participating media and reflective and refractive objects. The improvement of rendering performance is achieved without restrictions on the light paths or the scale of scenes. Moreover, lighting rays can undergo an arbitrary number of specular interactions as they travel through the scene, and the visual quality of radiance estimation is not limited by the resolution of a volume. As a result, various lighting features such as light glows, volumetric shadows and caustics can be swiftly generated for scenes of high complexity.

The central idea of our work is to evaluate single scattering only at the intersection points between lighting rays and viewing rays. Such computation can be significantly accelerated by constructing a spatial hierarchy in the parametric line space of Plücker coordinates for the lighting rays. Though an increase from 3D to 6D in the line representation incurs additional processing costs, our overall solution greatly reduces the storage and computation load in comparison to volumetric photon mapping, while maintaining high rendering quality. With the rapid development of GPU computation, we expect our technique to soon be useable in real-time applications.

In future work, there are a few issues we intend to examine. We plan to investigate methods for constructing a hierarchy based on line segments instead of entire lines, which would elevate the efficiency of our technique. In addition, we will experiment with adaptive kernels in the nearest neighbor search with the aim of improving rendering quality in areas with low light ray density. Finally, we plan to seek extensions of our technique for efficient rendering of multiple scattering and inhomogeneous participating media.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, and Shuitian Yan for artistry support. Kun Zhou was partially funded by the NSF of China (No. 60825201).

References

BASRI, R., HASSNER, T., AND ZELNIK-MANOR, L. 2009. A general framework for approximate nearest subspace search. In *IEEE Int. Workshop on Subspace Methods*.

BITTNER, J. 2003. *Hierarchical Techniques for Visibility Computations*. PhD thesis, Department of Computer Science and Engineering, Czech Technical University.

CHEN, M., AND ARVO, J. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph. 19*, 4, 246–278.

ERNST, M., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Interactive rendering of caustics using interpolated warped volumes. In *Proc. Graphics Interface (GI)*, 87–96.

HAVRAN, V., BITTNER, J., HERZOG, R., AND SEIDEL, H.-P. 2005. Ray maps for global illumination. In *Eurographics Symposium on Rendering*, 43–54.

HOU, Q., SUN, X., ZHOU, K., LAUTERBACH, C., MANOCHA, D., AND GUO, B. 2009. Memory–scalable gpu spatial hierarchy construction. Tech. rep., Microsoft Research Asia.

HU, W., DONG, Z., IHRKE, I., GROSCH, T., YUAN, G., AND SEIDEL, H.-P. 2010. Interactive volume caustics in single-scattering media. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, 109–117.

IHRKE, I., ZIEGLER, G., TEVS, A., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2007. Eikonal rendering: efficient light transport in refractive objects. *ACM Trans. Graph. 26*, 3, 59.

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2002. An efficient method for rendering underwater optical effects using graphics hardware. *Computer Graph. Forum 21*, 4, 701–711.

JAROSZ, W., DONNER, C., ZWICKER, M., AND JENSEN, H. W. 2008. Radiance caching for participating media. *ACM Trans. Graph. 27*, 1, 1–11.

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. The Beam Radiance Estimate for Volumetric Photon Mapping. *Computer Graph. Forum 27*, 2, 557–566.

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

KAJIYA, J. T. 1986. The rendering equation. In *Proc. ACM SIGGRAPH*, 143–150.

KRÜGER, J., BÜRGER, K., AND WESTERMANN, R. 2006. Interactive screen–space accurate photon tracing on GPUs. In *Rendering Techniques (Eurogr. Symp. Rendering – EGSR)*, 319–329.

MAX, N. L. 1986. Atmospheric illumination and shadows. In *Proc. ACM SIGGRAPH*, 117–124.

MITCHELL, D., AND HANRAHAN, P. 1992. Illumination from curved reflectors. In *Proc. ACM SIGGRAPH*, 283–291.

MOON, J. T., AND MARSCHNER, S. R. 2006. Simulating multiple scattering in hair using a photon mapping approach. *ACM Trans. Graph. 25*, 3, 1067–1074.

PAPADOPOULOS, C., AND PAPAIOANNOU, G. 2009. Realistic real–time underwater caustics and godrays. In *Proc. GraphiCon*, 89–95.

REN, Z., ZHOU, K., LIN, S., AND GUO, B. 2008. Gradient–based interpolation and sampling for real–time rendering of inhomogeneous, single–scattering media. *Computer Graph. Forum 27*, 7, 1945–1953.

SHREINER, D., WOO, M., NEIDER, J., AND DAVIS, T. 2005. *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison–Wesley Professional.

STOLFI, J. 1988. *Primitives for computational geometry*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, USA.

SUN, B., RAMAMOORTHI, R., NARASIMHAN, S. G., AND NAYAR, S. K. 2005. A practical analytic single scattering model for real time rendering. *ACM Trans. Graph. 24*, 3, 1040–1049.

SUN, X., ZHOU, K., STOLLNITZ, E., SHI, J., AND GUO, B. 2008. Interactive relighting of dynamic refractive objects. *ACM Trans. Graph. 27*, 3, 1–9.

TELLER, S. 1992. Computing the antipenumbra of an area light source. In *Computer Graphics*, 139–148.

WALTER, B., ZHAO, S., HOLZSCHUCH, N., AND BALA, K. 2009. Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph. 28*, 3, 1–8.

WYMAN, C., AND RAMSEY, S. 2008. Interactive volumetric shadows in participating media with single–scattering. In *IEEE Symp. Interactive Ray Tracing (IRT)*, 87–92.

ZHOU, K., HOU, Q., GONG, M., SNYDER, J., GUO, B., AND SHUM, H.-Y. 2007. Fogshop: Real–time design and rendering of inhomogeneous, single–scattering media. In *Proc. Pacific Conf. Comp. Graph. Appl. (PG)*, 116–125.

ZHOU, K., HOU, Q., WANG, R., AND GUO, B. 2008. Real–time kd–tree construction on graphics hardware. *ACM Trans. Graph. 27*, 5, 1–11.

ZHOU, K., REN, Z., LIN, S., BAO, H., GUO, B., AND SHUM, H.-Y. 2008. Real–time smoke rendering using compensated ray marching. *ACM Trans. Graph. 27*, 3, 1–12.