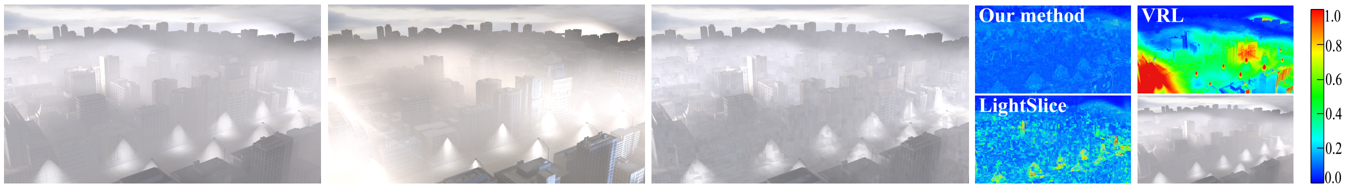
Adaptive Matrix Column Sampling and Completion for Rendering Participating Media

Yuchi Huo Rui Wang* Tianlei Hu Wei Hua Hujun Bao*

State Key Lab of CAD&CG, Zhejiang University



(a) *Our method* **(b)** *VRL* **(c)** *Adaptive LightSlice* **(d)** *Error maps and Reference.*

Figure 1: *Comparison of our method, VRL [Novák et al. 2012b], and adaptive LightSlice [Frederickx et al. 2015] on a city scene with heterogeneous smoke. Our method generates a high-quality result in approximately 20 min, whereas the other two algorithms produce results with severe artifacts at equal rendering time.*

Abstract

Several scalable many-light rendering methods have been proposed recently for the efficient computation of global illumination. However, gathering contributions of virtual lights in participating media remains an inefficient and time-consuming task. In this paper, we present a novel sparse sampling and reconstruction method to accelerate the gathering step of the many-light rendering for participating media. Our technique explores the observation that the scattered lightings are usually locally coherent and of low rank even in heterogeneous media. In particular, we first introduce a matrix formation with light segments as columns and eye ray segments as rows, and formulate the gathering step into a matrix sampling and reconstruction problem. We then propose an adaptive matrix column sampling and completion algorithm to efficiently reconstruct the matrix by only sampling a small number of elements. Experimental results show that our approach greatly improves the performance, and obtains up to one order of magnitude speedup compared with other state-of-the-art methods of many-light rendering for participating media.

Keywords: participating media, many-light rendering, matrix completion, adaptive rendering

Concepts: •**Computing methodologies** → **Rendering;**

1 Introduction

Participating media, such as clouds, fogs, liquid, and transparent solid, significantly contribute to a wide variety of vivid visual effects. Different kinds of rendering methods have been proposed and developed based on the radiative transfer equation (RTE) [Chandrasekhar 1960] to simulate such effects. However, one of the greatest challenges to solve this equation is the tremendous amount of possible light transport and scattering among particles, which inspires a series of solutions [Lafortune and Willems 1996; Georgiev et al. 2013; Jarosz et al. 2011b; Novák et al. 2012b; Novák et al. 2012a; Frederickx et al. 2015].

Many-light rendering is one class of methods having attracted increasing attention and achieved significant progress. These methods simplify the rendering into two steps, namely, virtual light shooting and virtual light gathering, as well as utilize the direct illuminations of the virtual lights to approximate multiple bounces of light reflection and scattering [Keller 1997; Raab et al. 2008; Engelhardt et al. 2012; Novák et al. 2012b]. Different types of virtual lights have been proposed and utilized to simulate different scattered lightings, such as points, rays, and beams. However, even when state-of-the-art representations, such as virtual ray lights (VRLs) or virtual beam lights (VBLs), and the most recent scalable approach [Frederickx et al. 2015] are used, gathering the contributions from a large number of virtual lights remains a tedious and time-consuming task.

In this paper, we introduce a new sparse sampling and reconstruction scheme for rendering participating media. Our method is based on the observation that the scattered lightings are usually locally coherent and of low rank even in heterogeneous participating media. By formulating the gathering step in participating media as a matrix sampling and reconstruction problem, we exploit the local coherence and low rank properties of the matrix, and apply the latest low-rank matrix completion technique [Krishnamurthy and Singh 2014] to accelerate the gathering from a small number of samples. Specifically, we first partition the VRLs and eye rays into segments and pair them to construct the lighting matrix. The matrix columns correspond to the segments of VRLs, and the rows correspond to the segments of eye rays. A novel adaptive sampling and reconstruction approach is then proposed to recover the matrix by sparsely sampling and completing columns. The results show that our method significantly improves the performance of rendering participating media, and obtains up to one order of magnitude speedup compared with state-of-the-art methods of many-light rendering for participating media.

The main contributions of this work are as follows:

- a new many-light rendering formation based on the segments of VRLs and eye rays for rendering participating media;

*Corresponding authors: Rui Wang (rwang@cad.zju.edu.cn), Hujun Bao (bao@cad.zju.edu.cn)

SA '16 Technical Papers,, December 05 - 08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12. . . $15.00
DOI: http://dx.doi.org/10.1145/2980179.2980244

- a new graph-partitioning algorithm to group and refine VRL and eye ray segments;
- a new adaptive matrix column sampling and completion method to recover the lighting matrix from a small number of elements; and
- an efficient rendering algorithm that heavily accelerates state-of-the-art methods and is capable of rendering a large amount of VRLs.

2 Related Work and Background

Many-light Rendering for Participating Media Numerous methods have been proposed to study the rendering of participating media. We recommend a survey [Pegoraro 2009] for a comprehensive introduction. In this paper, we focus only on relevant works, i.e., the many-light rendering approaches for participating media.

Many-light rendering originates from instant radiosity (IR) [Keller 1997], a special case of bidirectional path tracing, in which photons are first traced from light sources and stored as virtual point lights (VPLs), and indirect illuminations are gathered and approximated by these direct illuminations of VPLs. The reuse of light sub-paths of the IR method exhibits better performance. However, as the number of VPLs increases to thousands, millions, or more, efficiently gathering their contributions becomes a challenge, and motivates a series of scalable algorithms that are known as many-light rendering. Dachsbacher et al. [2014] provided a comprehensive survey on this class of methods.

Lightcuts [Walter et al. 2005] is a pioneer scalable solution that uses tree cuts from light tree to gather contributions from a large number of VPLs. Bidirectional Lightcuts [Walter et al. 2012] uses multiple bounce shading points to handle a high range of glossy material. Georgiev et al. [2012] sampled VPLs with sophisticated PDF to capture complex illumination. Matrix row and column sampling [Hašan et al. 2007] formulates the gathering of many lights as a problem of filling a lighting matrix. LightSlice [Ou and Pellacini 2011] clusters the rows of the lighting matrix into local slices and selects different representative VPLs to better capture subtle local lightings. Recently, a new matrix sampling-and-recovery method [Huo et al. 2015] has been proposed to recover the visibility of sub-lighting matrices with sparsely sampled matrix elements. Our work is inspired by this method, but solves a more challenging problem, i.e., rendering participating media.

Multidimensional Lightcuts [Walter et al. 2006] extends the idea of Lightcuts to high dimensional light trees; thus, this method can gather the scattered illuminations with a massive amount of VPLs in the media. However, the point-based representation suffers from either serious singularity artifacts or energy loss because of clamping. Raab et al. [2008] and Engelhardt et al. [2012] combined the traditional point-based IR with local path tracing to compensate the lost clamping energy. Nevertheless, the full compensation reverts to brute-force path tracing. Lately, new methods [Jarosz et al. 2011a; Jarosz et al. 2011b; Křivánek et al. 2014; Novák et al. 2012b; Novák et al. 2012a] extend the representation of scattering illumination from points to rays or beams, which significantly diminish the singularity artifacts. To represent virtual light, our method uses the VRL [Novák et al. 2012b] rather than VBL [Novák et al. 2012a] because the progressive scheme of the latter is more difficult to integrate in our matrix formation and sampling, although VBLs can better suppress the singularity artifacts than VRLs can.

Ray or beam-based approaches are more efficient in gathering virtual lights than point-based approaches. However, they still require tremendous sampling and integrations to render complex scenes with a large number, e.g., millions, of virtual lights. Fredericks et al. [2015] adapted LightSlice [Ou and Pellacini 2011] to accelerate the gathering in such a case. Different from their work, our method proposes a new matrix formation and introduces a new matrix sampling and reconstruction technique that completes the matrix only from a small number of matrix elements.

Adaptive Sampling The strategy of adaptive sampling has been widely used in rendering, especially in reducing the noise of Monte Carlo path tracing. Following the classical work [Mitchell 1987] in this field, adaptive sampling has achieved significant advances in recent years. Comprehensive reviews were conducted by Sen et al. [2015] and Zwicker et al. [2015]. In many-light rendering, numerous methods employ the idea of adaptivity. Lightcuts can be regarded as a generalized adaptive sampling algorithm [Walter et al. 2005; Walter et al. 2006; Walter et al. 2012; Frederickx et al. 2015], where it builds tree hierarchies of light sub-paths and adaptively connects them with eye sub-paths while considering a lower-error bound metric. In this paper, our method also possesses adaptivity by sampling matrix columns and completing the sampled columns adaptively.

Matrix Completion Matrix completion has recently become a popular topic in the research community. Theoretically, matrix completion aims to recover a low-rank matrix $\boldsymbol{L}$ from a set of sparsely sampled matrix elements by minimizing the matrix rank [Candès and Recht 2009]. The matrix rank of $\boldsymbol{L}$ is numerically approximated by its nuclear norm, defined as the sum of its singular values. Instead of completing the entire matrix at once [Candès and Recht 2009], Krishnamurthy et al. [2014] proposed an adaptive matrix completion method that recovers the matrix columns one by one by projecting each column into a low-rank sub-space of the matrix. We adopt this method in our approach because it column-wisely completes the matrix, so that we only need to sample and complete a small part of columns and then estimate the integration of all columns.

Matrix sampling and recovery techniques have also been used in recent rendering works. Huang et al. [2010] sparsely sampled and recovered light transport for precomputed rendering. Matrix row and sampling [Hašan et al. 2007] and its following work, LightSlice [Ou and Pellacini 2011], samples a subset of columns and rows of the lighting matrix to fill up the matrix. Wang et al. [2009] proposed a kernel Nyström method to reconstruct the lighting transport matrix from a few images. Peers et al. [2009] adopted compressive sensing techniques to reconstruct light transport for relighting. Ren et al. [2013] proposed to precompute radiance regression functions for real-time global illumination. Huo et al. [2015] sparsely sampled and recovered the visibilities of the lighting matrix. These methods accelerate the rendering of certain scenes but are unable to handle participating media.

3 Overview

In this section, we first present our matrix formation of rendering participating media and then provide an overview of our adaptive matrix column sampling and completion algorithm.

3.1 Rendering Participating Media with Virtual Lights

In our work, we follow the mathematical formation presented by Dachsbacher et al. [2014]. Theoretically, to render the participating media, the outgoing radiance at shading point $\boldsymbol{x}$ toward direction $-\boldsymbol{\omega}$ can be computed by gathering illuminations from VPLs as:

$$L_o(\boldsymbol{x}, -\boldsymbol{\omega}) = \sum_j I_j f(\boldsymbol{x}) G(\boldsymbol{x}, \boldsymbol{y}_j) \hat{V}(\boldsymbol{x}, \boldsymbol{y}_j), \quad (1)$$

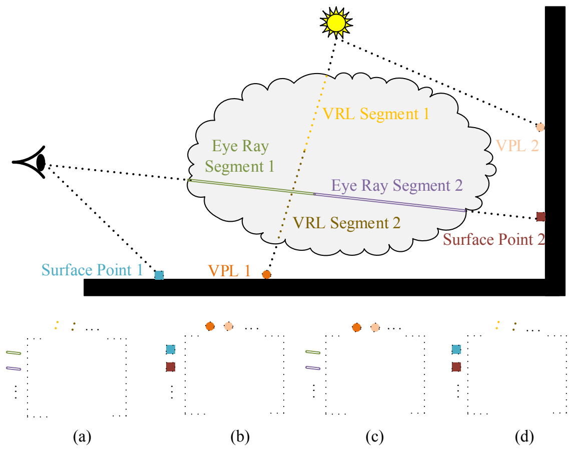


Figure 2: *Four types of sub-lighting matrices: (a) VRL segment to eye ray segment, (b) VPL to surface point, (c) VPL to eye ray segment and (d) VRL segment to surface point.*

where $\boldsymbol{y}_j$ and I_j are the position and intensity of j-th VPL, respectively; $f(\boldsymbol{x})$, $G(\boldsymbol{x}, \boldsymbol{y}_j)$ and $\hat{V}(\boldsymbol{x}, \boldsymbol{y}_j)$ are the generalized scattering distribution function (BRDF of the surface point or product of the phase function and scattering coefficient of media), generalized geometry term, and generalized visibility (product of normal visibility and transmittance), respectively. The detailed definitions of these functions can be found in the work of Dachsbacher et al. [2014].

To diminish the singularity produced by VPLs, Novák et al. [2012b] proposed to use eye rays and VRLs to replace shading points and VPLs, as well as integrated contributions between two rays as:

$$L_m \approx \int_0^{s'} \int_0^{t'} \tau(\boldsymbol{a}, \boldsymbol{x}_m) I_j \tau(\boldsymbol{b}, \boldsymbol{y}_m) \\ f(\boldsymbol{x}_m) f(\boldsymbol{y}_m) G(\boldsymbol{x}_m, \boldsymbol{y}_m) \hat{V}(\boldsymbol{x}_m, \boldsymbol{y}_m) dt ds, \tag{2}$$

where $\boldsymbol{x}_m = \boldsymbol{a} + t\boldsymbol{\omega}$ defines the eye ray starting from $\boldsymbol{a}$ to direction ω; $\boldsymbol{y}_m = \boldsymbol{b} + s\boldsymbol{\psi}$ defines the light ray starting from $\boldsymbol{b}$ to direction $\boldsymbol{\psi}$; $\tau(,)$ is the transmittance between two points; $f(\boldsymbol{x}_m)$ and $f(\boldsymbol{y}_m)$ are the generalized scattering distribution functions starting from $\boldsymbol{x}_m$ toward $\boldsymbol{y}_m$ and vice versa; and I_j is the intensity of j-th VRL.

3.2 Matrix Formation

Inspired by previous many-light rendering approaches [Hašan et al. 2007; Ou and Pellacini 2011; Huo et al. 2015; Frederickx et al. 2015], we formulate the rendering of participating media as the sampling and integrating of a lighting matrix. In particular, we split the eye rays and VRLs in the media into segments, and use the light transport between light and eye ray segments as the matrix element. By introducing these segments, we can formulate the entire rendering of the scene with participating media into four types of matrices, with two types of virtual lights as matrix columns, i.e., VRL segments and VPLs, as well as two types of receivers as matrix rows, i.e., shading points and eye ray segments. Examples of these matrices are illustrated in Figure 2. The matrix element in i row and j column is the contribution from j-th virtual light to i-th receiver. If the virtual light is a VRL segment and the receiver is an eye ray segment, then the matrix element of the segment-to-segment contribution is computed as:

$$E_{ij}^{ss} = \int_{s_j}^{s'_j} \int_{t_i}^{t'_i} \tau(\boldsymbol{a}_i, \boldsymbol{x}_m) I_j \tau(\boldsymbol{b}_j, \boldsymbol{y}_m) \\ f(\boldsymbol{x}_m) f(\boldsymbol{y}_m) G(\boldsymbol{x}_m, \boldsymbol{y}_m) \hat{V}(\boldsymbol{x}_m, \boldsymbol{y}_m) dt ds, \tag{3}$$

where $\boldsymbol{x}_m = \boldsymbol{a}_i + t\boldsymbol{\omega}$ and $\boldsymbol{y}_m = \boldsymbol{b}_j + s\boldsymbol{\psi}$ are the segments of VRL and eye ray, respectively; $\boldsymbol{a}_i$ is the origin of eye ray segment i; and $\boldsymbol{b}_j$ is the origin of VRL segment j.

Similarly, the other three types of matrix elements can be computed as:

$$E_{ij}^{ps} = \int_{t_i}^{t'_i} \tau(\boldsymbol{a}_i, \boldsymbol{x}_m) I_j f(\boldsymbol{x}_m) G(\boldsymbol{x}_m, \boldsymbol{y}_j) \hat{V}(\boldsymbol{x}_m, \boldsymbol{y}_j) dt, \tag{4}$$

$$E_{ij}^{sp} = \int_{s_j}^{s'_j} I_j \tau(\boldsymbol{b}_j, \boldsymbol{y}_m) f(\boldsymbol{x}_i) G(\boldsymbol{x}_i, \boldsymbol{y}_m) \hat{V}(\boldsymbol{x}_i, \boldsymbol{y}_m) ds, \tag{5}$$

$$E_{ij}^{pp} = I_j f(\boldsymbol{x}_i) G(\boldsymbol{x}_i, \boldsymbol{y}_j) \hat{V}(\boldsymbol{x}_i, \boldsymbol{y}_j), \tag{6}$$

where E_{ij}^{ps}, E_{ij}^{sp} and E_{ij}^{pp} are the point-to-segment, segment-to-point and point-to-point contributions from j-th virtual light to i-th receiver, respectively; $\boldsymbol{y}_j$ and $\boldsymbol{x}_i$ are the VPL and surface shading point, respectively.

3.3 Algorithm Overview

To calculate the final illumination of every row (receiver) of a sub-lighting matrix, we need to integrate all columns as follows:

$$\boldsymbol{s} = \sum_j \boldsymbol{l}_j, \tag{7}$$

where $\boldsymbol{l}_j$ denotes the j-th column of the sub-lighting matrix and the rows of $\boldsymbol{s}$ are the illuminations of the receivers. For a large scene with thousands or millions of virtual lights, accurately sampling and integrating every element of the lighting matrix are time-consuming and impractical. Therefore, we present a new adaptive matrix column sampling and completion algorithm to efficiently compute the final illuminations.

Based on the lighting matrix formations, three types of lighting matrices are composed of segments in rows, columns, or both. We use the matrix of eye ray and VRL segments as an example and show an overview of our algorithm in Figure 3. For other types of lighting matrices, these steps are similar but use points instead of segments. Our algorithm starts with the eye rays and VRLs generated by previous approaches [Novák et al. 2012b]. The algorithm then takes two steps, namely, matrix construction (Section 4) and adaptive column sampling and completion (Section 5). The basic idea is to first use the locality and coherence of the scattered lighting in media by slicing the entire lighting matrix on rows and building a hierarchy of columns, and then adaptively sample and complete columns in sliced sub-matrices. We take different strategies to explore the coherence of rows and columns. Compared with the number of eye rays, the number of virtual lights may vary dramatically in different scenes. Therefore, our algorithm slices the lighting matrix at first but postpones the selection of columns at the sampling step rather than at the matrix construction step to be scalable to these lights.

Matrix Construction Eye rays and VRLs are first divided into segments, which are clustered into groups and refined locally. We propose a similarity metric on segments and a graph-based partitioning algorithm to perform the segment clustering. Then, these clustered eye ray segments are used to slice the entire light matrix into sub-matrices, and the clustered VRL segments are used to construct a light tree. These sub-matrices with hierarchical columns are processed individually in the following step.

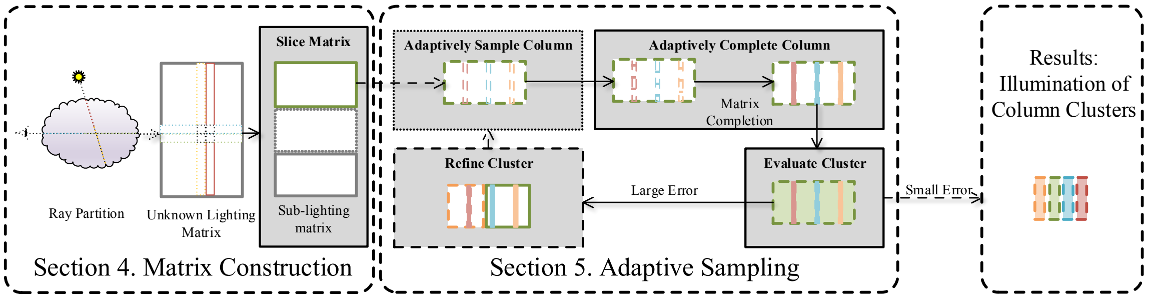


Figure 3: *Algorithm overview.*

Adaptive Column Sampling and Completion We adaptively sample only a few columns instead of all columns, and these columns are used to estimate the entire column integration. Furthermore, we subsequently employ an adaptive matrix completion technique that recovers these sampled columns by evaluating only a few elements of rows. In summary, our scheme samples a subset of the matrix columns and rows, thereby remarkably reducing the sampling cost.

4 Matrix Construction

In this section, we describe the algorithm to construct the lighting matrix. For simplicity, we focus on the construction of the segment-to-segment lighting matrix. Other types of lighting matrices can be constructed similarly if their rows or columns are ray segments.

Theoretically, the lower the rank of the matrix is, the fewer samples are required to complete it [Candès and Recht 2009; Krishnamurthy and Singh 2014]. Therefore, the goal of matrix construction is to obtain a lighting matrix with data coherence and low rank. Inspired by LightSlice [Ou and Pellacini 2011], we group the eye ray segments into clusters and slice the entire lighting matrix into sub-matrices. Furthermore, inspired by Lightcuts [Walter et al. 2005], we hierarchically group VRL segments into a light tree. These two operations on eye ray and VRL segments require a similarity metric to measure the difference in segments as well as a clustering algorithm to group similar ones. In the following, we first introduce the generation of an initial set of segments, then present a graph-based partitioning algorithm to group these segments, and finally describe the step to slice a matrix by eye ray segments and build the hierarchy on VRL segments.

4.1 Generating Initial Segments

Once the rays are obtained, we first generate a set of initial ray segments by distributing h points along each ray to partition it into $h+1$ segments, where $h = 16$ is set as default. We use Woodcock tracking to generate samples proportional to the transmittance to spread more samples in dense medium regions, which have more visual contributions to the image. A 2D VRL case is shown in Figure 4(a). However, even considering the importance of medium properties to generate points, these randomly distributed points may still fail to capture the local coherence of lighting transport among rays. Therefore, these segments are only regarded as an initial set of segments and will be refined later.

4.2 Graph-based Partitioning Algorithm

Given the initial set of segments, we propose a graph-based partitioning algorithm to group them into similar clusters and refine these ray segments using cluster boundaries. We first introduce the graph representation, then describe the partition algorithm, and finally present the step to refine the segments.

4.2.1 Graph Representation

We organize the segments into a graph, where the graph node is the segment and the edge weight is the similarity between two segments. The eye ray and VRL segments are separately organized into two graphs. The similarity defined on two segments, o_i and o_j, is computed as:

$$w_e(o_i,o_j) = S(o_i,o_j)+D(o_i,o_j)+T(o_i,o_j)+P(o_i,o_j), \quad (8)$$
$$w_l(o_i,o_j) = S(o_i,o_j)+D(o_i,o_j)+T(o_i,o_j), \quad (9)$$

where w_e and w_l are the similarities of eye ray and VRL segments respectively; S is the spatial distance term; D is the directional scattering term; T is the transmittance term; and P is the pixel-filtering term. The spatial distance S is computed as:

$$S(o_i,o_j) = \eta(\max(0, B - \parallel \boldsymbol{x}_i^0 - \boldsymbol{x}_j^0 \parallel - \parallel \boldsymbol{x}_i^1 - \boldsymbol{x}_j^1 \parallel)), \quad (10)$$

where $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_i^1$ are the start and end points of o_i, respectively; $\boldsymbol{x}_j^0$ and $\boldsymbol{x}_j^1$ are the start and end points of o_j, respectively; B is 1/8 of the diagonal length of the scene bounding box; and $\eta = \frac{1}{B}$ is a weight to normalize the spatial distance. The directional term D depicts the directional scattering difference between two segments as:

$$D(o_i,o_j) = \kappa(2 - d(\boldsymbol{x}_i^0,\boldsymbol{x}_j^0) - d(\boldsymbol{x}_i^1,\boldsymbol{x}_j^1)),$$
$$d(\boldsymbol{x}_i,\boldsymbol{x}_j) = \max\left(\frac{p_{\boldsymbol{x}_i}(\theta^m_{\boldsymbol{x}_i}) - p_{\boldsymbol{x}_j}(\theta^m_{\boldsymbol{x}_i})}{p_{\boldsymbol{x}_i}(\theta^m_{\boldsymbol{x}_i})}, \frac{p_{\boldsymbol{x}_i}(\theta^m_{\boldsymbol{x}_j}) - p_{\boldsymbol{x}_j}(\theta^m_{\boldsymbol{x}_j})}{p_{\boldsymbol{x}_i}(\theta^m_{\boldsymbol{x}_j})}\right), \quad (11)$$

where $\kappa = 0.5$ is a weight to normalize the term; $p_{\boldsymbol{x}_i}$ and $p_{\boldsymbol{x}_j}$ are the phase functions at two points of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively; $\theta^m_{\boldsymbol{x}_i}$ and $\theta^m_{\boldsymbol{x}_j}$ are the angles in the wold space that produce maximum phase function values at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively; and $\max(,)$ returns the larger difference between these two phase functions. We use the maximum value difference of these two phase functions to depict the difference between them, because the computation is much cheaper. T distinguishes the energy attenuations caused by transmittance variations, especially in heterogeneous media, of two segments as:

$$T(o_i,o_j) = 1 - \|\tau(\boldsymbol{x}_i^0,\boldsymbol{x}_i^1) - \tau(\boldsymbol{x}_j^0,\boldsymbol{x}_j^1)\|, \quad (12)$$

of which the values can be directly computed from the cached transmittance along VRLs. The pixel filtering term is a Gaussian filter specifically defined for eye rays as:

$$P = 1 - \frac{1}{2\pi\sigma}e^{-\frac{\bar{x}^2+\bar{y}^2}{2\sigma^2}},$$

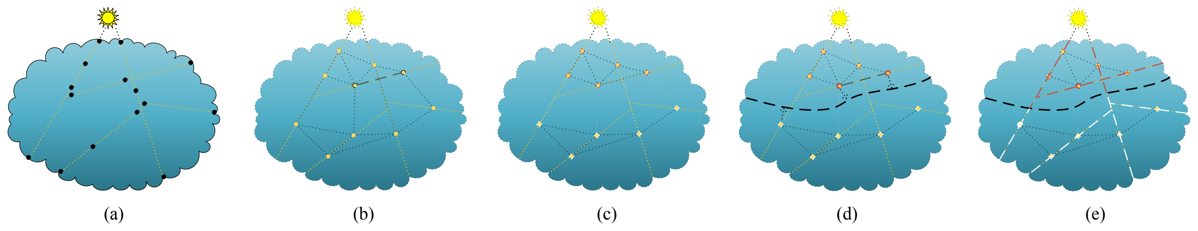


Figure 4: *Illustration of ray partitioning and clustering process. (a) Tracking sample points to generate initial segments. (b) Constructing a graph from segments, in which the graph is visualized by linking one point on each segment. (c) Partitioning the graph into red and white clusters. (d) Finding the cluster boundary. (e) Refining segments with the boundary.*

where $\bar{x}$ and $\bar{y}$ are the horizontal and vertical image distances between o_i and o_j, respectively; and σ is a parameter set to 4. We define this term in the image space to differentiate eye ray segments with close start points from the camera.

Ideally, for each pair of segments, an edge with similarity exists. However, considering that the spatial distance of two segments is large, the similarity distance becomes dominated by the spatial distance. Therefore, in practice, we consider only the spatial neighbors of segments. Each segment is linked to its nearest five spatial neighbors by searching the nearest start or end points of segments in a point KD-tree. Figure 4(b) illustrates a VRL segment graph, in which graph nodes are in orange dot and edges are in black lines.

4.2.2 Graph Partitioning

Given a graph, we convert the clustering problem into a graph-partitioning problem to partition graph nodes (segments) into groups. These groups are then used to slice the lighting matrix and build the VRL hierarchy, which is discussed in Section 4.2.3. We use the K-cut graph-partitioning algorithm to partition the graph into K groups while minimizing the sum of the edge weights that connect nodes in different groups. The accurate K-cut graph partitioning is a NP-hard problem, but with several efficient approximation algorithms. We adopt the mtMetis proposed by LaSalle et al. [2013]. Figure 4(c) illustrates an example, in which the nodes are partitioned into two groups that are denoted as red and white. We select the graph-partitioning algorithm because this approach effectively preserves segments of the same rays; therefore, the segments of different clusters are not interlaced with other rays. The K-means clustering may fail to preserve the spatial connection because it directly encompasses spatial information into a high-dimensional distance metric, in which spatial connections might be suppressed by other factors. We present further comparisons in Section 6.1.

4.2.3 Segment Refinement

The initial segments are randomly generated. Consequently, they may not effectively capture the local coherence of the media. We observe that these segment clusters, after the graph partitioning, expose global information on how segments should be distributed in the media. For example, the boundary among clusters reflects certain changes in the media. Therefore, these segments across the boundary should be divided into two segments, so that such changes among clusters can be better preserved and in-cluster coherence can be improved. We propose a three-step refinement procedure to utilize the underlying information of segment clusters and improve the segments along rays, i.e., find better start or end points of segments.

The first step is to determine the boundary between two neighboring clusters. We adopt the support vector machine (SVM) to find an optimal decision hyperplane between two clusters. We use only the start or end points of boundary segments as the training data in the SVM to simplify the computation. The radial basis function $K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\gamma^2}}$ with $\gamma = 0.5$ is used as a nonlinear kernel to compute the support vector, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are positions of the points. The nonlinear kernel enables us to compute the non-planar boundary between clusters. Given the boundary, we then split the segments across the boundary and create new segments. Finally, each newly created segment is compared with the connecting segment in the same cluster using the similarity metric (Eq.(8) or (9)). They are merged if they are similar enough, i.e., the similarity metric is larger than a threshold of 2.5.

4.3 Slicing Lighting Matrix and Building Light Tree

For VPLs or shading points, we use the methods proposed in previous studies [Ou and Pellacini 2011; Frederickx et al. 2015; Huo et al. 2015] to cluster rows and build a hierarchy of lights.

For eye rays, w e directly partition them into K clusters. K is empirically set to 0.008 times the total number of eye rays. With this setting, each group contains approximately 500 to 1000 segments (rows) after the partition. Such a setting experimentally suits our matrix completion algorithm in terms of efficiency. Upon obtaining the clusters of eye ray segments, we use them to slice the lighting matrix into sub-matrices.

For VRLs, we iteratively apply two-cut partitions on the VRL segments until all segments in the same node reach the fixed similarity threshold. At each iteration, a graph is partitioned into two subgraphs. These partitions result in a hierarchy of subgraphs, which directly maps to a virtual light tree.

5 Adaptive Sampling Algorithm

With sliced sub-lighting matrices and virtual light trees, the next step is to compute the final illumination of every receiver (point or segment) by sampling and integrating columns of each sub-lighting matrix. In this section, we introduce our sampling and reconstruction algorithm that adaptively samples columns and completes the sampled columns from a small number of elements.

The pseudo code of the entire algorithm is shown in Algorithm 1. The overall procedure of our algorithm exhibits certain similarities to the tree traversal scheme proposed by Lightcuts [Walter et al. 2005]. In particular, we hierarchically traverse the light tree to compute a cut. All tree nodes on the cut are stored in a priority queue, $Queue$, and sorted by errors. At each traversal, one node with the largest error is popped and split into children nodes, where each child node is sampled and evaluated in the $SampleAndEvaluate$ routine. Once the error of this child node is less than a threshold, we accumulate its contribution to receivers and stop the traversal on this node; otherwise, this child node is pushed into the queue.

However, our method fundamentally differs from the standard Lightcuts procedure in several aspects. First, in Lightcuts, a representative light is used to estimate the overall contribution of a

Algorithm 1 Adaptive Sampling Algorithm

```
function ADAPTIVE SAMPLING($tree$)
    $Queue \leftarrow \emptyset$
    $Queue$.PUSH($tree.root$)
    while $Queue \neq \emptyset$ do
        $node \leftarrow Queue$.POP()
        $left\_child, right\_child \leftarrow$ REFINE($node$)
        SAMPLEANDEVALUATE($left\_child$)
        SAMPLEANDEVALUATE($right\_child$)
        if $left\_child.clustering\_error > threshold$ then
            $Queue$.PUSH($left\_child$)
        else
            TERMINATE($left\_child$)
        end if
        if $right\_child.clustering\_error > threshold$ then
            $Queue$.PUSH($right\_child$)
        else
            TERMINATE($right\_child$)
        end if
    end while
end function

function SAMPLEANDEVALUATE($node$)
    $\{columns\} \leftarrow$SAMPLECOLUMN($node$)
    for each $column$ in $\{columns\}$ do
        COMPLETECOLUMN($column$)
    end for
    EVALUATENODEILLUMINATION($node$, $\{columns\}$)
    EVALUATECLUSTERINGERROR($node$, $\{columns\}$)
end function
```

node; in our method, constructing a representative light segment for segments with varying lengths, intensities and scattering coefficients is impractical. Thus, we propose a column sampling method to sample a subset of columns in the node and estimate the overall contribution from them. Second, while sampling the column, our method only uses a small number of rows to complete the entire column instead of fully sampling all rows. Finally, instead of using a conservative error bound, our method computes the error from a number of columns at a tree node because of the difficulty in estimating an error bound on VRLs. In the following subsections, we first describe the procedures to sample and complete columns and then introduce the step to evaluate the node error.

5.1 Computing Node Illumination

A tree node on the cut contains a set of light segments. Given a tree node and receivers represented in columns and rows of a submatrix, integrating of node illumination aims to compute all illuminations from light segments of the node to all rows. Instead of densely sampling all elements, we adopt the idea of adaptive matrix column completion [Krishnamurthy and Singh 2014] that utilizes the sub-column space to reconstruct columns by sampling only a few elements of rows. However, even when the sub-column space is used, still many columns have to be sampled remain. Instead, inspired by the idea proposed by Georgiev et al. [2010] to sample VPLs, we use the Monte Carlo integration that only randomly samples a small number of columns. In the following, we first introduce the column completion step and then describe the Monte Carlo integration step.

5.1.1 Column Completion

An adaptive matrix completion algorithm [Krishnamurthy and Singh 2014] is adapted for our column completion. Technically, we maintain a sub-column space $\boldsymbol{U}$ of the matrix to complete columns. We use $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ to denote a sub-lighting matrix and $\boldsymbol{U} \in \mathbb{R}^{m\times k}$

Algorithm 2 Column Completion Algorithm

```
1:  function MATRIXCOMPLETION($\boldsymbol{L}$)
2:      $\boldsymbol{U} \leftarrow \emptyset$
3:      $\Omega \leftarrow$ DRAWROWINDICES()
4:      for each column $\boldsymbol{l}$ of $\boldsymbol{L}$ to sample do
5:          $\boldsymbol{l}_\Omega \leftarrow$ DRAWROWSAMPLES($\boldsymbol{l}, \Omega$)
6:          if $\|\boldsymbol{l}_\Omega - \boldsymbol{U}_\Omega(\boldsymbol{U}_\Omega^T\boldsymbol{U}_\Omega)^{-1}\boldsymbol{U}_\Omega\boldsymbol{l}_\Omega\|_1 > threshold$ then
7:              densely sample $\boldsymbol{l}$ then add to $\boldsymbol{U}$ (orthogonalize $\boldsymbol{U}$)
8:              $\Omega \leftarrow$ DRAWROWINDICES()
9:          else
10:             $\bar{\boldsymbol{l}} \leftarrow \boldsymbol{U}(\boldsymbol{U}_\Omega^T\boldsymbol{U}_\Omega)^{-1}\boldsymbol{U}_\Omega^T\boldsymbol{l}_\Omega$
11:         end if
12:     end for
13: end function
```

to denote the sub-column space of $\boldsymbol{L}$, where $k \ll n$. Each column of $\boldsymbol{U}$ is one orthogonal basis of the sub-column space. When sampling a new column, $\boldsymbol{l}$, we first sample a few elements of this new column, where $\Omega \in \mathbb{R}^{\lfloor \rho m \rfloor}$ denotes uniformly sampled indices ranging from 0 to m, and $\boldsymbol{l}_\Omega \in \mathbb{R}^{\lfloor \rho m \rfloor}$ is the down-sampled version of $\boldsymbol{l}$, which contains only sampled rows, i.e. the rows indexed by Ω. Then, the entire column $\boldsymbol{l}$ can be reconstructed by projecting the down-sampled column back into the column space $\boldsymbol{U}$ as:

$$\boldsymbol{l} \approx \bar{\boldsymbol{l}} = \boldsymbol{U}(\boldsymbol{U}_\Omega^T\boldsymbol{U}_\Omega)^{-1}\boldsymbol{U}_\Omega^T\boldsymbol{l}_\Omega, \tag{13}$$

where $\boldsymbol{U}_\Omega \in \mathbb{R}^{\lfloor \rho m \rfloor \times k}$ is a down-sampled version of $\boldsymbol{U}$ that only contains rows indexed by Ω and removes other rows. If $\boldsymbol{L}$ is characterized by low rank and low matrix coherence [Candès and Recht 2009; Krishnamurthy and Singh 2014], then only a small ratio of randomly sampled elements is sufficient to recover the entire matrix, thereby reducing the computational cost of completing $\bar{\boldsymbol{l}}$.

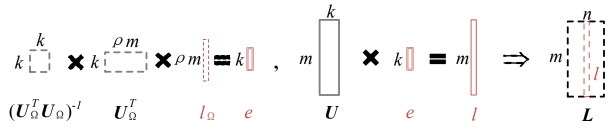


Figure 5: *Illustration of one column recovery.*

The pseudo code of this column completion algorithm is provided in Algorithm 2, and an illustration of the recovery of one column is shown in Figure 5. The column completion algorithm draws a small set of row indices from the routine $DrawRowIndices$ and places them in Ω. Rows of unknown columns are sampled according to the indices in Ω in the routine $DrawRowSamples$. In line 6, we test whether the down-sampled column $\boldsymbol{l}_\Omega$ is in the down-sampled space $\boldsymbol{U}_\Omega$. If this down-sampled column l_Ω is in the space, then it is directly projected back into the full column space $\boldsymbol{U}$ (line 10). Otherwise, the entire column $\boldsymbol{l}$ is sampled and added to the column space $\boldsymbol{U}$.

We use $E_\Omega(\boldsymbol{l}_\Omega)$ to represent the operation that projects a down-sampled column $\boldsymbol{l}_\Omega$ into the space $\boldsymbol{U}$ as $E_\Omega(\boldsymbol{l}_\Omega) = (\boldsymbol{U}_\Omega^T\boldsymbol{U}_\Omega)^{-1}\boldsymbol{U}_\Omega^T\boldsymbol{l}_\Omega$, and $\boldsymbol{e} \in \mathbb{R}^k = E_\Omega(\boldsymbol{l}_\Omega)$ to denote the projection of $\boldsymbol{l}_\Omega$ on the sub-column space $\boldsymbol{U}_\Omega$. Such a projection $\boldsymbol{e}$ can be used to recover the entire column $\boldsymbol{l}$ from the down-sampled column $\boldsymbol{l}_\Omega$ as $\boldsymbol{l} \approx \bar{\boldsymbol{l}} = \boldsymbol{U}\boldsymbol{e}$, where $\bar{\boldsymbol{l}}$ is the approximation of the original column $\boldsymbol{l}$. Using these projections and recoveries, we can efficiently perform the integration and error estimation in the sub-column space $\boldsymbol{U}$, in which the column size of sub-column space is much smaller than the original column space, $k \ll n$, and the dimension of the projection $\boldsymbol{e}$ is much smaller than the row size, $\rho m \ll m$.

5.1.2 Integrating Columns

Based on the sub-column space reconstruction, the integration of the entire sub-lighting matrix is the summation of all columns

$\sum_j \boldsymbol{l}^j \approx \boldsymbol{U} \sum_j \boldsymbol{e}^j$. By combining the idea of Monte Carlo sampling in VPLs [Georgiev and Slusallek 2010], we can compute the integration of the node as:

$$\boldsymbol{i}(node) \approx \boldsymbol{U}\boldsymbol{e}(node) = \boldsymbol{U}\frac{1}{K}\sum_{k=1}^{K}\frac{E_\Omega(\boldsymbol{l}_\Omega^k)}{PDF(\boldsymbol{l}_\Omega^k)}, \tag{14}$$

where K is the number of sampled columns in the node, $PDF(\boldsymbol{l}_\Omega^k)$ is the sampling probability of column k, $\boldsymbol{l}_\Omega^k$ is the randomly sampled column, $E_\Omega(\boldsymbol{l}_\Omega^k)$ calculates the projection of column $\boldsymbol{l}^k$ in space $\boldsymbol{U}$, and $\boldsymbol{e}(node)$ is the integration of projections. We simply use intensities of virtual lights as the PDF and sample 10 columns per node as default. More details can be found in the supplementary document. By taking Eq.(14) into Eq.(7), we compute the final illumination of the entire sub-lighting matrix by summing all contributions of nodes as:

$$\boldsymbol{s} = \sum \boldsymbol{U}\boldsymbol{e}(node) = \boldsymbol{U}\sum \boldsymbol{e}(node), \tag{15}$$

where, we first accumulate the projections in the sub-column space and then project them back to the full column space, which further reduce the computation of integration.

5.2 Estimating Node Errors

Two kinds of errors are used in our algorithm to control the procedures of sampling and reconstruction. The first error is the completion error in line 6 of Algorithm 2. This error represents the energy loss in the column completion. If one column $\boldsymbol{l}$ cannot be well reconstructed by sub-column space $\boldsymbol{U}$, then this column will be fully sampled. Such an error is computed as:

$$\frac{\|\boldsymbol{l}_\Omega - \boldsymbol{U}_\Omega \boldsymbol{e}\|_1}{\|\boldsymbol{l}_\Omega\|_1}\|\boldsymbol{e}\|_1 > \sigma\|\boldsymbol{e}_{all}\|_1, \tag{16}$$

where $\boldsymbol{e}_{all}$ is the accumulated contribution of all nodes in the current tree cut; σ is a predefined ratio parameter, with 0.05 as default; and $\frac{\|\boldsymbol{l}_\Omega - \boldsymbol{U}_\Omega \boldsymbol{e}\|_1}{\|\boldsymbol{l}_\Omega\|_1}$ estimates the percentage of energy loss for $\boldsymbol{l}$.

The second error is introduced to evaluate how well the node contribution is estimated by some of its columns (lines 9 and 14 of Algorithm 1). We use an error metric similar to the adaptive clustering metric proposed by Hachisuka et al. [2008]. Specifically, the column sampling error $\boldsymbol{r}(node)$ for $node$ is computed as:

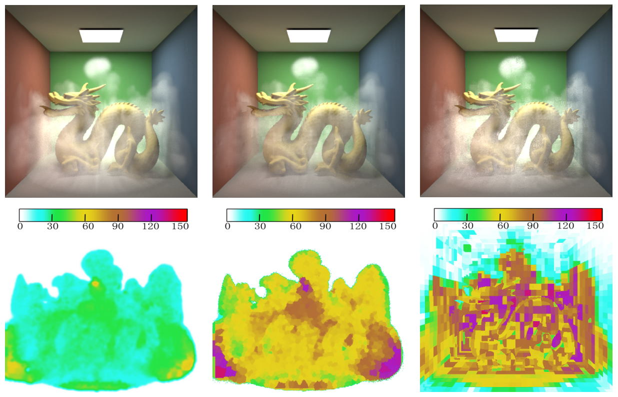
$$\boldsymbol{r}(node) = \frac{1}{N}\sum_{N}^{n}(\boldsymbol{e}(node) - \frac{E_\Omega(\boldsymbol{l}_\Omega^n)}{PDF(\boldsymbol{l}_\Omega^n)})^2. \tag{17}$$

The average squared difference between sampled columns and the estimated node contribution in space $\boldsymbol{U}$ is calculated. If $\boldsymbol{r}(node) > \beta\boldsymbol{e}_{all}$, then the estimate of this node is not good enough, and the node is split. We set $\beta = 0.01$ as default.

6 Results

We implement our method on a PC with two Intel Xeon E5-2630 CPUs and 32GB memory. The matrix computation is conducted using the Intel MKL library. Different virtual lights, including directional, omni and diffuse oriented VPLs [Walter et al. 2005] and standard VRLs [Novák et al. 2012b], are used in test scenes. We compare the quality and performance of our method with those of the original VRL [Novák et al. 2012b] and the recent adaptive LightSlice [Frederickx et al. 2015]. All algorithms are parallelized in our test machine, but the reference images are rendered by a cluster with 32 PCs in tens of hours. The statistics of demo scenes are listed in Table 2. The column sampling ratio is manually set as 0.1 in most cases. The maximum memory consumptions of our method, adaptive LightSlice and VRL are 470, 280 and 200 MB, respectively. Compared with previous methods, the additional memory used in our method is for storing the hierarchies of VPLs and segments. Unless mentioned specifically, we use the same original method of VRL [Novák et al. 2012b] to compute the segment-to-segment illuminations.

6.1 Comparison of Different Partitions of Rays



(a) *Our method* (b) *K-Means* (c) *No Partitioning*

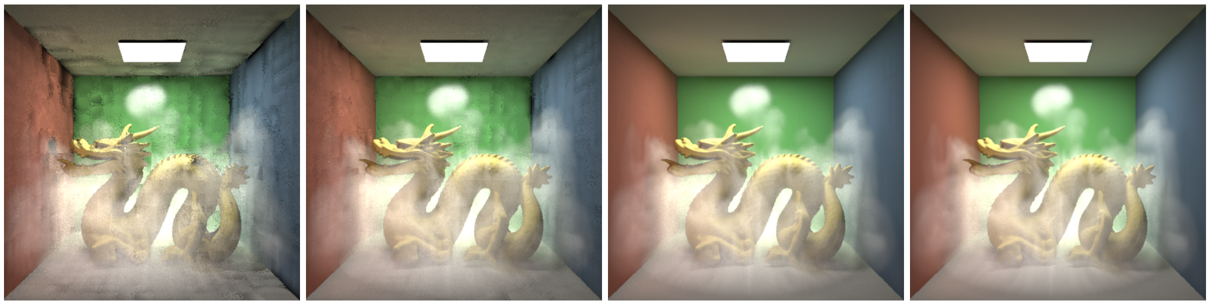
Figure 6: *Comparison of different partitions of rays. The eye rays are partitioned by three different strategies and rendered with equal sampling ratios. The reference image of this scene is shown in Figure 7(d).*

Figure 6 shows the results generated with three ray-partitioning strategies, namely, the proposed graph-based partitioning, K-Means clustering, and no partitioning on rays. The previous approaches [Frederickx et al. 2015] can be regarded as using no partitioning strategy. We visualize the ranks of sub-lighting matrices in Figure 6 (bottom). A lower rank indicates fewer samples to sample and complete these matrices. In Figure 6 (top), we render the scene with an equal sampling ratio. As can be seen, our graph-based partitioning obviously outperforms the other two ray partitioning strategies.

6.2 Results of Column Completion

We show several results that are only produced by column completion without adaptive column sampling (i.e, we fully sample every column but we only complete each column by a few row elements) to better evaluate the completion step in our algorithm. We first show the effect of the error threshold σ on the quality of the column completion algorithm in Figure 7. Large σ results in some loss of lighting details. In our practice, $\sigma = 0.005$ is good enough to recover subtle lighting effects; therefore, it is used as default in all our experiments.

We then use a studio scene to show the effect of the sampling ratio ρ on the quality of the column completion algorithm. In this scene, the major light source comes from the window and directly illuminates the smoke, in which the major illumination of the scene is contributed by indirect VRLs. We forcibly sample every column of the sub-lighting matrix, and test various sampling ratio ρ of rows. As shown in Figure 9, with only 10% of elements per column, the resultant image is close to the reference. After counting the overhead to conduct the column completion, our column completion achieves over 7 times speedup in generating an image with a similar quality. While the column sampling ratio decreases, blocky



(a) $\sigma = 0.05$ **(b)** $\sigma = 0.01$ **(c)** $\sigma = 0.005$ **(d)** *Reference*

Figure 7: *Comparisons of results using different error thresholds σ in the column completion. The rendering time for $\sigma = 0.05$, $\sigma = 0.01$ and $\sigma = 0.005$ are 40, 48, 52 s respectively.*

artifacts appear. Such artifacts appear first at the boundary of light shaft, where the rank is higher, and then in the entire image.

6.3 Results of Anisotropic Materials

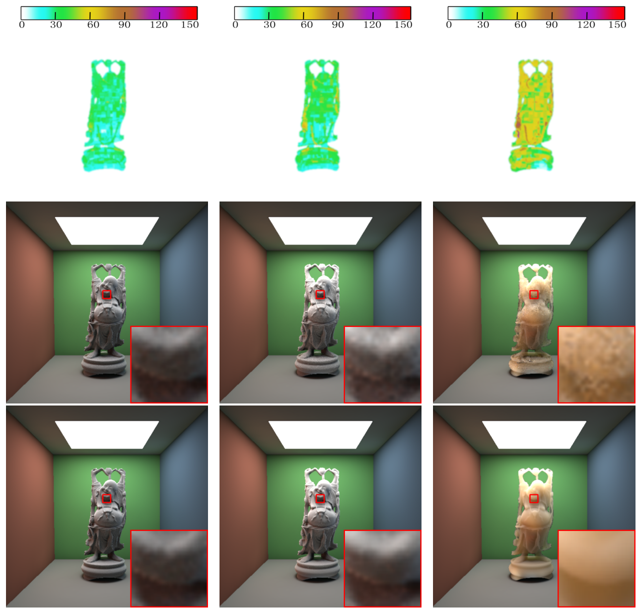
We apply our algorithm to render participating media with anisotropic materials, and show the results in Figure 8. From left to right, the Buddha models have different coefficients g of the Henyey-Greenstein (HG) phase function, ranging from purely isotropic 0 to highly anisotropic 0.99. At the top row of Figure 8, the ranks of sub-lighting matrices are presented, in which the rank increases as the anisotropic coefficient increases. We use two sampling ratios, 0.1 and 0.2, to render the scene. As shown in the middle and bottom rows of Figure 8, when the material becomes increasingly anisotropic, the lower sampling ratio tends to produce more noticeable artifacts because of the lack of samples. The increase rank of the lighting matrix leads to the increase of sampling ratio. Therefore, if the scene is complex both in lighting and material, our algorithm might degrade to the traditional full sampling algorithm.

6.4 Final Results and Comparison

We compare the overall performance of our method with those of VRLs [Novák et al. 2012b] and adaptive LightSlice [Frederickx et al. 2015] by rendering various scenes at equal time. Error images show the relative L^1 absolute errors. Specifically, $e = \frac{\|L_o - L\|_1}{L_o}$, where L_o is the intensity of the reference image, and L is the intensity of the test image. We also plot the curves of time versus number of virtual lights for different methods to better compare the scalability of different methods.

The city scene in Figure 1 is the largest scene in our paper. The entire city is covered by heterogeneous smoke and lit by sky lighting from the sun and several street lamps. The sun and the lamps produce light shafts in the smoke. The complex effects require a large number of virtual lights to capture the lighting. At equal time, the VRL method generates serious artifacts because of only a small number of VRLs that can be processed during a limited time. For the same reason, the adaptive LightSlice has some obvious blocky artifacts, especially in the regions lit by the lamps. By contrast, our algorithm works well in this scene because our method samples 10% of the elements per column. In this way, our method can process more columns in the same time.

The Sponza scene in Figure 10 is a small but commonly used scene to test the complex indirect lighting transport in the corridor. At the same processing time, only a small number of virtual lights coming into the yard are captured by the VRL method. As a result, this method produces a large light shaft and spiky lighting artifacts. The adaptive LightSlice captures most lighting features of the scene but still has blocky artifacts as shown in the error map. Our method generates a smoother result at the same processing time. The curves on the bottom right of Figure 10 show the scalabilities of the three methods under the same error threshold. For one millions virtual lights, our method outperforms the adaptive LightSlice over one order of magnitude because of our sparse sampling and completion scheme.



(a) $g = 0$ **(b)** $g = 0.5$ **(c)** $g = 0.99$

Figure 8: *Buddha model with different anisotropic coefficients. The scattering and absorption coefficients are set to $(1, 1, 1)$ and $(0.002, 0.005, 001)$, respectively, for all models. However, the anisotropic coefficient g of HG phase functions differs. The top row visualizes the rank. The middle and bottom rows are rendered with sampling ratios of $\rho = 0.1$ and $\rho = 0.2$, respectively. The time needed to render models with $g = 0$, $g = 0.5$ and $g = 0.99$ at $\rho = 0.2$ is 62, 68 and 75 s, respectively.*

Finally, we test the three methods on a laboratory scene with various participating media, as shown in Figure 11. From left to right, the media are soft drink (Gatorade), jade, soap, juice (grapefruit), skin and milk with different HG phase functions. These material parameters are from previous studies [Gkioulekas et al. 2013b; Jakob 2010; Gkioulekas et al. 2013a] and listed in Table 1. These media comprise anisotropic media (such as jade, beverage and soap) and

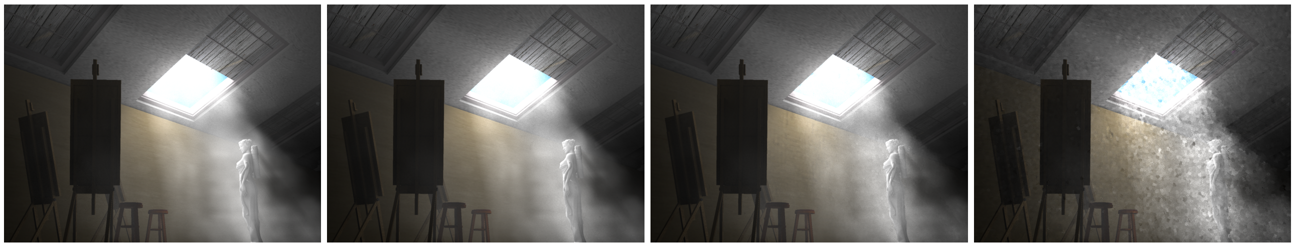

(a) *Reference* **(b)** $\rho = 0.1$ **(c)** $\rho = 0.05$ **(d)** $\rho = 0.01$

Figure 9: *Studio scene.*

	σ_s	σ_a	g
Drink	0.04,0.04,0.06	0.02,0.02,0.009	0.93,0.93,0.94
Jade	1,1,1	0.002,0.004,0.007	0.9,0.9,0.9
Soap	0.2,0.2,0.22	0.001,0.001,0.002	0.96,0.95,0.94
Juice	0.15,0.15,0.16	0.03,0.17,0.48	0.93,0.93,0.93
Skin	0.74,0.88,1	0.03,0.17,0.48	0,0,0
Milk	1,1,1	0.0026,0.005,0.01	0.82,0.8,0.75

Table 1: *Media parameters in the Lab scene.*

thick media (such as soap, skin and jade). As shown in the results, with the anisotropic media, our method still produces an image with a much better quality than those of the VRL and adaptive LightSlice within the same time period. The VRL method has obvious singularities and the adaptive LightSlice has blocky artifacts, especially in the shadow regions. By contrast, our method generates smoother results. However, because of anisotropic scattering, our method requires a higher sampling ratio of 0.2 in column completion to capture the high-frequency light transport and has less speedup than that in the Sponza scene.

In summary, under equal time comparison, the quality of our method is better than those of VRL and adaptive LightSlice. With more virtual lights being processed in the equal time, our method can generate smoother and more accurate results. As a result of the adaptive sampling of columns, the cost of our method is sublinear to the number of virtual lights. In our test scenes, we gain up to two orders of magnitude speedup compared with the VRL [Novák et al. 2012b] and one order of magnitude speedup compared with the adaptive LightSlice [Frederickx et al. 2015].

7 Conclusion and Future Work

In this paper, we present a new adaptive matrix column sampling and completion method to accelerate the rendering of participating media. Our method formulates the final gathering of virtual lights, especially gathering VRLs to eye rays, into a matrix formation with segment-to-segment contributions. VRLs and eye rays in the media are partitioned into local segments and clustered to form sub-lighting matrices and a light tree. We design an adaptive matrix column sampling and completion algorithm to efficiently compute the integration along columns. This algorithm utilizes the local coherence and low-rank property of the sub-lighting matrix by sampling only a small number of elements of the matrix to gather contributions from virtual lights. Such a sampling and reconstruction strategy can be regarded as performing sparse sampling at two dimensions, namely, along columns and rows. In columns, we sample only a subset of columns to reconstruct the integration of all columns; for each column, only a small number of rows are sampled to reconstruct the entire column through column completion. Adaptive sampling in these two dimensions effectively exploits the low rank of the lighting matrix and significantly accelerates the gathering from the VRLs.

Our method, however, has several limitations. First, the sampling ratio increases with the rank of the lighting matrix. Therefore, our method might need a higher sampling ratio to render scenes with a high frequency of light scatterings. Second, we cannot automatically detect the best sampling ratio for different scenes or different regions of the scene. Adaptively estimating the optimal sampling ratios for different matrices is a promising future direction, but it remains challenge in the research of matrix completion. Third, similar to many other many-light methods, our method does not explicitly support temporal coherence. Preserving temporal coherence involves a series of works that are beyond the scope of our paper, such as ways to distribute virtual point lights to diminish flickering [Dachsbacher et al. 2014]. Addressing this limitation is a highly challenging and interesting future task to explore the potential of our method in time domain.

The proposed algorithm is a general framework for other effects that can be formulated as matrix integration problems, including extending this method to a higher dimension, such as motion blur, depth of field, or temporal coherent animation. Applying the adaptive matrix column sampling and completion or the sparse sampling concept to other rendering methods, such as bidirectional volumetric path tracing, and even problems outside rendering, is worth exploring in future.

Acknowledgement

We would like to thank Kavita Bala for the initial discussion and all reviewers for their thoughtful comments. We also would like to thank Shihao Jin and Haohua Liu for preparing demos. This work was partially supported by National Key R&D Program of China (No. 2016YFB1001503), NSFC (No. 61472350), the Fundamental Research Funds for the Central Universities (No. 2016FZA5013).

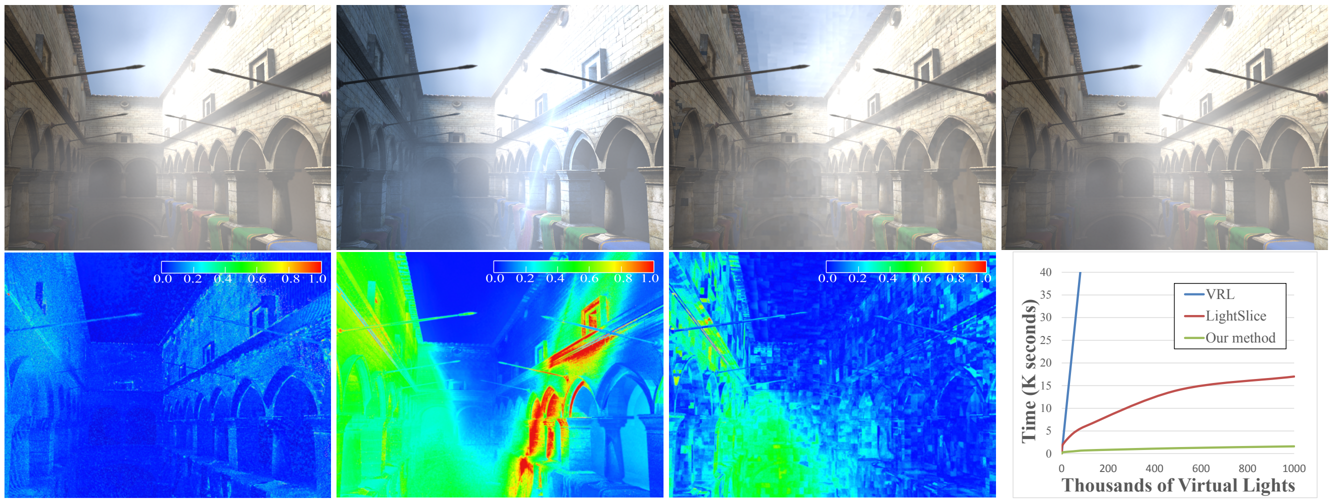
References

CANDÈS, E., AND RECHT, B. 2009. Exact matrix completion via convex optimization. *Found. Comput. Math. 9*, 6 (Dec.), 717–772.

CHANDRASEKHAR, S. 1960. *Radiative transfer*. Courier Corporation.

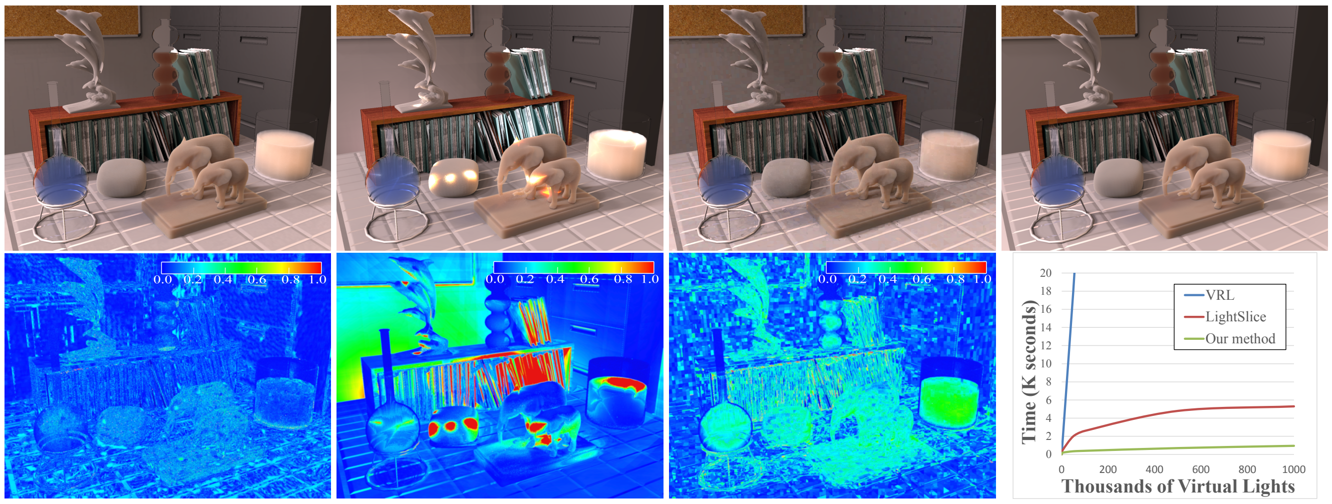
DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., AND NOVÁK, J. 2014. Scalable realistic rendering with many-light methods. In *Computer Graphics Forum*, vol. 33, Wiley Online Library, 88–104.

ENGELHARDT, T., NOVK, J., SCHMIDT, T., AND DACHSBACHER, C. 2012. Approximate bias compensation for rendering scenes with heterogeneous participating media. In *Computer Graphics Forum*, 2145–2154.



(a) *Our method* (b) *VRL(equal-time)* (c) *Lightslice(equal-time)* (c) *Reference and Time/VL Curve*

Figure 10: *Sponza scene*



(a) *Our method* (b) *VRL(equal-time)* (c) *Lightslice(equal-time)* (c) *Reference and Time/VL Curve*

Figure 11: *Lab scene*

FREDERICKX, R., BARTELS, P., AND DUTRÉ, P. 2015. Adaptive lightslice for virtual ray lights. In *EUROGRAPHICS 2015-Short Papers*, The Eurographics Association, 61–64.

GEORGIEV, I., AND SLUSALLEK, P. 2010. Simple and robust iterative importance sampling of virtual point lights. In *In Eurographics short papers*.

GEORGIEV, I., KŘIVÁNEK, J., POPOV, S., AND SLUSALLEK, P. 2012. Importance caching for complex illumination. *Computer Graphics Forum 31*, 2. EUROGRAPHICS 2012.

GEORGIEV, I., KRIVANEK, J., HACHISUKA, T., NOWROUZEZAHRAI, D., AND JAROSZ, W. 2013. Joint importance sampling of low-order volumetric scattering. *ACM Trans. Graph. 32*, 6, 164–1.

GKIOULEKAS, I., XIAO, B., ZHAO, S., ADELSON, E. H., ZICKLER, T., AND BALA, K. 2013. Understanding the role of phase function in translucent appearance. *ACM Transactions on Graphics (TOG) 32*, 5, 147.

GKIOULEKAS, I., ZHAO, S., BALA, K., ZICKLER, T., AND LEVIN, A. 2013. Inverse volume rendering with material dictionaries. *Acm Transactions on Graphics 32*, 6, 1210–1214.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. In *ACM Transactions on Graphics (TOG)*, vol. 27, ACM, 33.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26*, 3, 26:1–10.

HUANG, F.-C., AND RAMAMOORTHI, R. 2010. Sparsely precomputing the light transport matrix for real-time rendering. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 1335–1345.

HUO, Y., WANG, R., JIN, S., LIU, X., AND BAO, H. 2015. A matrix sampling-and-recovery approach for many-lights rendering. *ACM Trans. Graph. 34*, 6 (Oct.), 210:1–210:12.

JAKOB, W., 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

JAROSZ, W., NOWROUZEZAHRAI, D., SADEGHI, I., AND JENSEN, H. W. 2011. A comprehensive theory of volumet-

		sponza	city	lab	studio($\rho = 0.1$)
scene	triangles	2.9K	153K	107K	346K
	resolution	1600×1200	1920×1080	1600×1200	1600×1200
	VPLs/VRLs	723K/490K	890K/1.5M	1M/572K	547/608
	ray partition time(s)	14	21	18	11
eye rays	column sample ratio	0.1	0.1	0.2	0.1
	avg. densely sampled columns	28	35	42	22
	avg. sparsely sampled columns	3689	3180	2652	1133
	avg. sampled tree nodes	385	342	274	-
	column completion time(s)	31	36	22	12
	other time (s)	761	803	462	255
surface points	column sample ratio	0.1	0.1	0.2	0.1
	avg. densely sampled columns	31	35	26	14
	avg. sparsely sampled columns	2873	3152	2135	1141
	avg. sampled tree nodes	290	340	257	-
	column computation time(s)	55	47	37	15
	other time (s)	357	451	319	175
summary	avg. relative error	5.5%	5.8%	6.4%	2.6%
	total time (s)	1218	1337	840	457
equal-time VRL	avg. relative error	27%	33.5%	24.8%	-
(no clustering)	VPLs/VRLs	927/510	604/1076	759/436	-
equal-time Lightslice	avg. relative error	14.2%	17.5%	18.2%	-

Table 2: *The statistics of the scenes and results. The surface points and eye rays are separately counted, on which the column completion time is the pure algebraic computation of matrix completion algorithm. The last two sections are equal-time statistics of the VRL and the adaptive Lightslice.*

ric radiance estimation using photon points and beams. *ACM Transactions on Graphics (TOG) 30*, 1, 5.

JAROSZ, W., NOWROUZEZAHRAI, D., THOMAS, R., SLOAN, P.-P., AND ZWICKER, M. 2011. Progressive photon beams. *ACM Transactions on Graphics (TOG) 30*, 6, 181.

KELLER, A. 1997. Instant radiosity. In *Proc. SIGGRAPH '97*, 49–56.

KRISHNAMURTHY, A., AND SINGH, A. 2014. On the power of adaptivity in matrix completion and approximation. *Eprint Arxiv*.

KRIVÁNEK, J., GEORGIEV, I., HACHISUKA, T., VÉVODA, P., SIK, M., NOWROUZEZAHRAI, D., AND JAROSZ, W. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph. 33*, 4, 103–1.

LAFORTUNE, E., AND WILLEMS, Y. 1996. Rendering participating media with bidirecitonal path tracing. In *Eurographics Workshop on Rendering*, 91–100.

LASALLE, D., AND KARYPIS, G. 2013. Multi-threaded graph partitioning. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, IEEE, 225–236.

MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. In *ACM SIGGRAPH Computer Graphics*, vol. 21, ACM, 65–72.

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012. Progressive virtual beam lights. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 1407–1413.

NOVÁK, J., NOWROUZEZAHRAI, D., DACHSBACHER, C., AND JAROSZ, W. 2012. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph. 31*, 4, 60:1–60:11.

OU, J., AND PELLACINI, F. 2011. Lightslice: matrix slice sampling for the many-lights problem. *ACM Trans. Graph. 30*, 6 (Dec.), 179:1–179:8.

PEERS, P., MAHAJAN, D. K., LAMOND, B., GHOSH, A., MATUSIK, W., RAMAMOORTHI, R., AND DEBEVEC, P. 2009. Compressive light transport sensing. *ACM Transactions on Graphics (TOG) 28*, 1, 3.

PEGORARO, V. 2009. *Efficient physically-based simulation of light transport in participating media*. University of Utah.

RAAB, M., SEIBERT, D., AND KELLER, A. 2008. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 591–605.

REN, P., WANG, J., GONG, M., LIN, S., TONG, X., AND GUO, B. 2013. Global illumination with radiance regression functions. *ACM Transactions on Graphics (TOG) 32*, 4, 130.

SEN, P., ZWICKER, M., ROUSSELLE, F., YOON, S.-E., AND KALANTARI, N. K. 2015. Denoising your monte carlo renders: recent advances in image-space adaptive sampling and reconstruction. In *ACM SIGGRAPH 2015 Courses*, ACM, 11.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. 24*, 3, 1098–1107.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. In *ACM Transactions on Graphics (TOG)*, vol. 25, ACM, 1081–1088.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Trans. Graph. 31*, 4 (July), 59:1–59:11.

WANG, J., DONG, Y., TONG, X., LIN, Z., AND GUO, B. 2009. Kernel nyström method for light transport. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 29.

ZWICKER, M., JAROSZ, W., LEHTINEN, J., MOON, B., RAMAMOORTHI, R., ROUSSELLE, F., SEN, P., SOLER, C., AND YOON, S.-E. 2015. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. In *Computer Graphics Forum*, vol. 34, Wiley Online Library, 667–681.