Modular Flux Transfer: Efficient Rendering of High-Resolution Volumes with Repeated Structures

Shuang Zhao
Cornell University

Miloš Hašan
Autodesk

Ravi Ramamoorthi
University of California, Berkeley

Kavita Bala
Cornell University

Figure 1: *We introduce a modular flux transfer (MFT) framework to approximate high-order scatterings in extremely complex volumes.* ***Left:*** *a scene containing a purple tablecloth with* 2009×3300 *yarn crossings represented by an anisotropic volume consisting of* 1.06×10^{13} *effective voxels.* ***Center:*** *a* $5\times$ *zoomed version of the left image, illustrating the complexity of the cloth volume.* ***Right:*** *a* $25\times$ *zoomed version.*

Abstract

The highest fidelity images to date of complex materials like cloth use extremely high-resolution volumetric models. However, rendering such complex volumetric media is expensive, with brute-force path tracing often the only viable solution. Fortunately, common volumetric materials (fabrics, finished wood, synthesized solid textures) are structured, with repeated patterns approximated by tiling a small number of exemplar blocks. In this paper, we introduce a precomputation-based rendering approach for such volumetric media with repeated structures based on a modular transfer formulation. We model each exemplar block as a voxel grid and precompute voxel-to-voxel, patch-to-patch, and patch-to-voxel flux transfer matrices. At render time, when blocks are tiled to produce a high-resolution volume, we accurately compute low-order scattering, with modular flux transfer used to approximate higher-order scattering. We achieve speedups of up to $12\times$ over path tracing on extremely complex volumes, with minimal loss of quality. In addition, we demonstrate that our approach outperforms photon mapping on these materials.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

Keywords: rendering, precomputation, textiles

Links: DL PDF WEB

ACM Reference Format
Zhao, S., Hašan, M., Ramamoorthi, R., Bala, K. 2013. Modular Flux Transfer: Efficient Rendering of High-Resolution Volumes with Repeated Structures. ACM Trans. Graph. 32, 4, Article 131 (July 2013), 11 pages. DOI = 10.1145/2461912.2461938 http://doi.acm.org/10.1145/2461912.2461938.

DOI: http://doi.acm.org/10.1145/2461912.2461938

1 Introduction

High-quality rendering of complex materials increasingly uses volumetric data, such as the high-resolution micro-CT cloth models developed by Zhao et al. [2011; 2012]. However, rendering optically dense volumetric media is challenging for multiple reasons: their high resolution (with voxel sizes of a few microns), the complex occlusion in the medium, and the anisotropic phase functions that influence light scattering. Moreover, computation is dominated by multiple scattering within long light paths (5-100 scattering events), especially when the single-scattering albedo is high. This leads to an undesirable situation, where rendering brighter colored materials becomes substantially more expensive. To date, pure Monte Carlo path tracing has proven the only reliable method for rendering such materials, but so far has been too slow for widespread use: tens to hundreds of core hours are required to produce the images in [Zhao et al. 2011].

Our goal is to significantly improve upon this situation; the approach we take is based on the following insights. First, complicated volumetric media like cloth often contain repetitive building blocks, such as yarn crossings. This suggests the possibility of precomputing light transport in these blocks, and *modularly* combining them into a complex volume: we are inspired by [Loos et al. 2011] who introduced a similar idea for approximating indirect lighting in blocked scenes. Second, in dense media with high albedos, such as white or bright-colored fabrics, high-order multiple scattering is the most expensive component and contributes significantly to overall appearance [Moon et al. 2008; Jakob et al. 2010]. This suggests that speedup is most likely to be achieved by accelerating the computation of the extremely expensive, but lower-frequency multiple scattering. A similar approach is common for subsurface scattering, but diffusion approximations do not easily apply to the complex volumetric media we are interested in, since they are often based on assumptions of isotropy, homogeneity, and flat boundaries.

The key challenge to a scalable solution is high dimensionality. In the most general case, light transport is a linear operator that maps emitted radiance into final radiance. The radiance on the boundary of a volume is a function of position and direction: a 4-dimensional light field. Worse yet, a precomputed approach seems to require the 8-dimensional linear mapping between two such light fields. Our key insight is that we can handle the curse of dimensionality

inherent in this problem by slightly modifying the long light paths, while keeping the short light paths intact. This way we make the problem tractable while maintaining accuracy.

We use these insights to develop a precomputation-based method for rendering volumetric media with repeated structures. Our algorithm separates low-order and high-order scattering, and modularly precomputes the latter. We model the volumetric medium as a grid of *voxelized blocks*. While a full volume may consist of millions of blocks, there are only a few (tens or hundreds) unique *exemplar* blocks required to synthesize it [Zhao et al. 2012]. We precompute the voxel-to-voxel, patch-to-patch, and patch-to-voxel flux transfer matrices for each of these exemplar blocks. At render time, we modularly stitch together light transport using the precomputed matrices to efficiently compute higher-order scattering. Our contributions include:

- A new formulation for the efficient and tractable rendering of anisotropic volumes by exploiting modularity, and avoiding the curse of dimensionality.
- A Monte Carlo matrix inversion based algorithm to make our approach tractable for very large volumes with millions of blocks, each with thousands of voxels.
- Results demonstrated on a variety of highly complicated volumes, with a focus on cloth (Figures 1, 14, and 15), but generalizing to non-cloth synthesized volumes (Figure 16) as well.

We show that this approach can, in many cases, accelerate rendering over path tracing by an order of magnitude. It also significantly outperforms techniques like photon mapping on these media. While the previous research on highly complex measured volumetric materials generated significant interest, it was too expensive in practice. We believe that our method, which runs fast enough on a single server with relatively low memory requirements, will make these materials usable in demanding industry applications including interior design and digital effects.

2 Related Work

Volumetric path tracing. Solution of the radiative transfer equation (RTE) by Monte Carlo methods was first introduced to computer graphics by Kajiya and von Herzen [1984], and has since frequently been used to render participating media such as clouds or fog. More recently, volumetric path tracing has also been applied to the rendering of fabrics. Jakob et al. [2010] introduced an anisotropic version of the RTE with several desirable properties of reciprocity and energy conservation, and the microflake phase function, especially well suited for cloth rendering.

Zhao et al. [2011] measured the volume densities of cloth samples using micro-CT scanning, which are then rendered using the microflake model. While requiring slow rendering times, this work shows that high-quality microgeometry makes a significant visual difference. In follow-up work, Zhao et al. [2012] synthesized larger patterns from micro-CT data with trillions of effective voxels, making rendering even more expensive. Bidirectional lightcuts [Walter et al. 2012] decreases the noise in volumetric path tracing, but achieves efficiency gains only under complex illumination for such data.

Diffusion approximations. For many materials, expensive Monte Carlo rendering of volumetric light transport can be effectively approximated by cheaper methods. Jensen et al. [2001] introduce a dipole model to graphics using diffusion theory for rendering subsurface scattering materials. However, this work uses the homogeneous isotropic assumption, in addition to assuming a semi-infinite flat slab geometry; all of these assumptions are false for fabrics. Donner and Jensen [2007] remove the flat slab assumption by placing multiple-scattering point sources along a light ray through the medium; the problem in applying this idea to fabrics is that the fluence distribution created by a point source inside cloth is far from symmetric. The layered subsurface scattering approach of Donner and Jensen [2005] is also modular, but only handles flat layers. D'Eon [2011] introduces a more accurate modification of diffusion theory, but it does not easily apply to our volumes with strong heterogeneity and anisotropic scattering.

An alternative to analytic point source derivations is to solve the diffusion equation by finite differences [Wang et al. 2008] or finite elements [Arbree et al. 2011]. These methods only apply to slowly-changing isotropic media with a flat refractive boundary. The work by Jakob et al. [2010] also introduces an extended diffusion approximation for anisotropic media, but this does not easily apply to fabrics, which would require billions of simple finite elements. In contrast, our method's complex "elements" (blocks) contain a sizable portion of the cloth with a significant number of scattering events. In general, volumes containing fully anisotropic scattering from fibers, empty space, shadowing and interreflections defy an analytic approach.

Methods approximating higher bounces. Several existing solutions share our approach of splitting out the first few path segments, and then looking up a radiance approximation. Volumetric photon mapping [Jensen and Christensen 1998], also extended to anisotropic scattering from blond hair by Moon and Marschner [2006], approximates radiance by using density estimation after a few Monte Carlo bounces. The problem with density estimation for rendering high-resolution microgeometries is that to get enough photons of all necessary orientations and path lengths, the radius of lookup has to be orders of magnitude larger than the underlying microgeometry, thus being locally inaccurate. Photon beams and the beam radiance estimate [Jarosz et al. 2011] can be very useful in optically thin media; however, the mean free path in cloth is a few microns, so the usable length of a beam would likely become much smaller in our case.

The follow-up works by Moon et al. [2007; 2008], and the related hair rendering approach of Zinke et al. [2008], are also based on using approximate radiance after several path-traced bounces. However, these methods do not take advantage of the modularity from repeated structures, and may not be scalable to the complexity of our volumes. Schroeder et al [2011] render fabrics by replacing actual microgeometry for higher bounces by randomly selected and oriented fibers; however, the required path lengths remain the same, and the evaluation of the model is also quite expensive.

Precomputed approaches. Modular radiance transfer [Loos et al. 2011] was recently introduced for diffuse indirect illumination for blocked interiors often found in computer games, and targeted at real-time performance. The key idea is that light needs to be propagated from a surface to a block boundary, then within blocks, and finally back to a surface. We take significant inspiration from this approach, and show how to derive a modular formulation in the domain of high-quality volume rendering.

The Lumislice approach [Xu et al. 2001] precomputes scattering within a yarn of cloth, and produces convincing results for fabrics with thick yarns, but does not easily extend to fabrics modeled at the fiber level, or strong anisotropic highlights. Lensch et al. [2003] precompute heterogeneous subsurface scattering for a mesh, and compress the transport matrices using a local-global decomposition. A similar approach could be useful to further compress the precomputed transfer matrices in our method. Premože et al. [2004] proposed an approach to compute multiple scattering with a "light attenuation volume" precomputed for each light source. This method focuses on thin media like fog and cannot capture highly anisotropic scattering.

Radiosity approaches. Several techniques (like [Xu et al. 1990; Arnaldi et al. 1994; Lewis and Fournier 1996]) share the idea of partitioning complex scenes into smaller ones (which is related to our modular transfer approach) but apply only to surface-based rendering. Furthermore, stochastic radiosity [Bekaert 1999] is highly

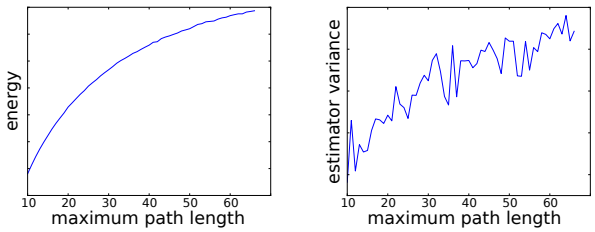


Figure 2: *The increase in the total energy of an image, and the corresponding increase in the variance of a Monte Carlo path tracer, as a function of the maximum number of scattering events (with number of samples held constant). The data is measured on our felt scene (top of Figure 14), where the green single-scattering albedo has been set to 0.99.*

related to our Monte Carlo matrix inversion, but focuses on avoiding the $O(n^2)$ cost of the form factor matrix, while we crucially need to eliminate even the $O(n)$ cost per voxel.

Other methods. Rushmeier and Torrance [1987] integrated volume scattering in thin participating media into a radiosity framework. This technique formulates light transport using finite elements, related to our approach for a single block, but is not modular and does not scale to high-resolution volumes since block-level precomputation is impossible. Narasimhan et al. [2003] derived a general formula to solve the RTE. Their formula, however, applies only to homogeneous participating media and thus cannot be used to solve our problem. Fattal [2009] presented an approach based on the Discrete Ordinates Method to solve the RTE approximately. This technique is able to render heterogeneous materials like marble, but is not efficient enough to handle volumes with trillions of voxels.

Path formulation of volume rendering. A powerful path formulation of the rendering equation was introduced by Veach [1997], and extended by Pauly et al. [2000] to volumetric media. The intensity of a pixel is expressed as an integral over all light paths in the scene passing through this pixel.

$$I = \int_\Omega f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{1}$$

Here μ is a measure on the path space $\Omega = \bigcup_{l\geq 1}\Omega_l$, and Ω_l is the space of paths with l segments, $\bar{x} = \mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_l$, such that $\mathbf{x}_0$ is the camera position, $\mathbf{x}_1$ is the surface point or volumetric scattering location directly visible through the pixel, $\mathbf{x}_l$ is on a light source and $\mathbf{x}_2,\ldots,\mathbf{x}_{l-1}$ are any light bounce points in the scene.

The contribution $f(\bar{x})$ of any single path $\bar{x} = \mathbf{x}_0\mathbf{x}_1\ldots\mathbf{x}_l$ is the product of a *pixel weight term* $W(\mathbf{x}_0\leftarrow\mathbf{x}_1)$, a *light emission* term $L_e(\mathbf{x}_{l-1}\leftarrow\mathbf{x}_l)$, *geometry terms* $G(\mathbf{x}_k\leftrightarrow\mathbf{x}_{k+1})$ corresponding to each segment of the path, and *material terms* $M(\mathbf{x}_{k-1}\leftarrow\mathbf{x}_k\leftarrow\mathbf{x}_{k+1})$ corresponding to every interior vertex. Geometry terms contain exponential extinction (inside volumes) and $1/r^2$ falloff when necessary. Note also that cosine terms are included in G for surface but not volume events. In the case of volumetric media, the material terms contain a (possibly anisotropic) phase function evaluation:

$$M(\mathbf{x},\boldsymbol{\omega}_1,\boldsymbol{\omega}_2) = \alpha(\mathbf{x})\,\sigma_t(\mathbf{x},\boldsymbol{\omega}_1)\,p(\mathbf{x},\boldsymbol{\omega}_1,\boldsymbol{\omega}_2) \tag{2}$$

We assume the albedo $\alpha(\mathbf{x})$ is not directionally dependent. The material term is reciprocal, though the phase function is generally not; as detailed by Jakob et al. [2010], the phase function observes the slightly more involved reciprocity relationship $\sigma_t(\mathbf{x},\boldsymbol{\omega}_1)\,p(\mathbf{x},\boldsymbol{\omega}_1,\boldsymbol{\omega}_2) = \sigma_t(\mathbf{x},\boldsymbol{\omega}_2)\,p(\mathbf{x},\boldsymbol{\omega}_2,\boldsymbol{\omega}_1)$. We use the microflake phase function [Jakob et al. 2010; Zhao et al. 2011], which satisfies this relationship, in addition to other desirable properties.

$\bar{x}$	a light path (sequence of vertices)
$\mu(\bar{x})$	measure on path space (product of surface and volume measures on vertices)
$f(\bar{x})$	product of geometry and material terms along path, pixel weight (for camera vertices) and light emission (for light vertices)
$G(\mathbf{x}_1\leftrightarrow\mathbf{x}_2)$	geometry term (contains $1/r^2$ falloff when necessary, volume extinction, and cosine terms for surface, but not volume vertices)
$M(\mathbf{x},\boldsymbol{\omega}_1,\boldsymbol{\omega}_2)$	material term: a product of the phase function $p(\mathbf{x},\boldsymbol{\omega}_1,\boldsymbol{\omega}_2)$, the single-scattering albedo $\alpha(\mathbf{x})$, and the extinction coefficient $\sigma_t(\mathbf{x},\boldsymbol{\omega}_1)$
B_i	block: a grid of voxels
V_i	voxel: at precomputation level, not data level
N_i	non-empty subset of voxel V_i
$\lvert N_i\rvert$	volume of N_i
P_i	patch: an oriented voxel face on the shared interface between two blocks
$\mathrm{flip}(i)$	flip operator on set of patch indices
Q	a permutation matrix encoding $\mathrm{flip}(i)$
T_i^{vv}, T_i^{vp}, T_i^{pv}, T_i^{pp}	voxel-to-voxel, voxel-to-patch, patch-to-voxel, and patch-to-patch transfer matrices for block B_i
$\tilde{T}^{vv}$, $\tilde{T}^{vp}$, $\tilde{T}^{pv}$, $\tilde{T}^{pp}$	global completions of transfer matrices
$\boldsymbol{\Phi}^s$	source flux: integral of illumination up to first scattering event
$\boldsymbol{\Phi}^{gt}$	ground truth multiple-scattered flux
$\boldsymbol{\Phi}^m$	multiple-scattered flux (approximation of $\boldsymbol{\Phi}^{gt}$ computed by modular transfer)

Table 1: *Summary of notation; bold letters indicate vectors.*

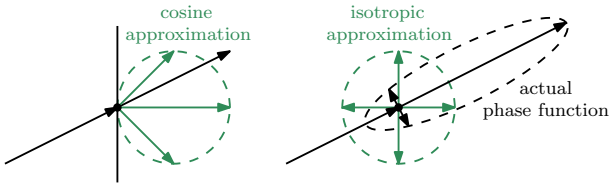
We use this path formulation to describe our method. Note that path integrals can also be easily applied to paths that do not have endpoints on the camera or light source.

3 Overview

In this section, we describe the main computational difficulty in rendering complex, optically thick media with anisotropic phase functions. We introduce two key simplifications to make the problem tractable: isotropy and diffuser events in long light paths. Based on these simplifications, we introduce a modular approach, which precomputes the light transport within blocks of the volume. The modularity is inspired by Loos et al.'s [2011] approach. However, there are differences in the main challenges that need to be handled. While Loos et al.'s main challenge is compression of the transport into very small vectors of coefficients that allow for real-time evaluation, we instead need to deal with extremely large resolution and long light paths.

Key challenge: long paths. The volumetric path integral can be naturally evaluated by Monte Carlo sampling of paths with known probabilities, which leads to a standard volumetric path tracer. The problem with this approach is its slow performance, especially for materials with highly complicated structures, such as data-driven cloth, where creating each path vertex executes a Woodcock tracking procedure to importance-sample extinction, and has a significant cost of many non-cached memory accesses. Furthermore, a significant amount of energy (and resulting variance) is in paths with many segments. This problem is most significant for bright colors, which have a high single-scattering albedo in one or more color channels. Figure 2 shows the increase in energy in a cloth sample if albedo is high, and the corresponding increase in variance in a path tracing solution.

Diffuser and isotropy events. Shorter paths can be handled by pure path tracing; like some previous approaches, we apply our approximations to longer paths. The key difference is that we take advantage of the repeated structure in the volume to *pre-*



(a) diffuser event (b) isotropy event

Figure 3: *Inserting isotropy events and diffuser events into paths makes the underlying path integral separable, at the cost of introducing some error. If these events are inserted into paths of sufficient length, the error will be close to imperceptible.*

compute a large number of paths, driving the effective number of paths significantly higher (and the noise much lower) than any non-precomputed approach.

One key challenge is to split paths into precomputed and runtime components without introducing high-dimensional (and therefore expensive) intermediate data representations. The coupled nature of the material terms along a path through an anisotropic volume makes any attempt to split the problem into subproblems (some of which could be precomputed) seem hopeless.

We propose modifying the problem by inserting *diffuser events* and *isotropy events* into some paths (see Figure 3). We found that setting the phase function on a small number of vertices of a long path to be isotropic creates very little error in the final image, which makes precomputation feasible. Therefore, we insert isotropy events on the k-th vertex from the camera and the last vertex before the light. Also, we insert zero or more diffuser events in between. These modifications provide significant advantages by making the underlying path integrals separable, in the sense that they can be factored into a product of integrals "before" and "after" the isotropy/diffuser event. This allows us to split the paths into components that are precomputed offline, and computed at runtime. In other words, this enables us to use *flux* instead of *light fields* as the quantity being transported, addressing the curse of dimensionality, and making the problem tractable. These assumptions are later used in Sections 5.1 and 5.3. In comparison, modular radiance transfer [Loos et al. 2011] uses low-resolution lightfields at block boundaries. We considered coarse directional discretizations, but found their storage to be too expensive: for example, 32 directional bins would inflate our precomputed data 1,024 times.

We show that, if done for paths of sufficient length, these modifications have little effect on the accuracy of the solution. In fact, as our results show, the accuracy of our results is higher with a practical number of samples than with methods like path tracing or photon mapping. While theoretically these approaches do not make any isotropy or diffuser approximations, in practice they need many more samples to achieve a quality as high as our results (Section 6.2).

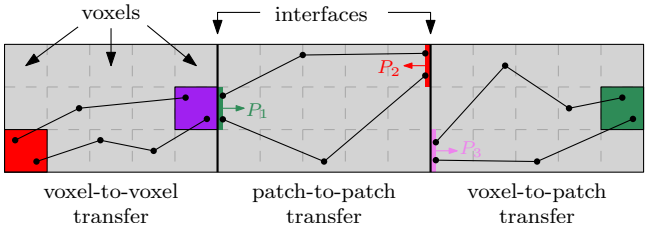


Figure 4: *Definitions of voxels, interfaces, patches, and three types of precomputed transfers, each of which corresponds to a matrix. Note that patch-to-voxel transport is the transpose of voxel-to-patch.*

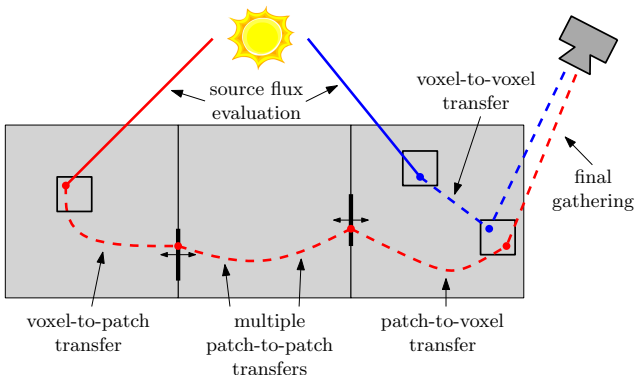


Figure 5: *The three phases of the runtime stage of our pipeline. We use dashed lines to indicate sub-paths with length ≥ 1 which can contain multiple scattering events.*

3.1 Definitions

Blocks. We divide the volume into a number of equal-sized *blocks*. Our algorithm makes no assumption about the definition of a block. For our woven fabric data, a block is defined to be of size about 0.5 mm (approximately a yarn crossing, as suggested in [Zhao et al. 2012]). We treat the volume as consisting of a 2-dimensional grid of blocks without loss of generality; the grid could also easily be 3-dimensional, but in our results there is always only one block across the thickness of the volume). The full volume can be made of millions of blocks, but due to repeating structures, only a small number (up to one or two hundreds in our case) of the blocks are unique, and we call them *exemplar blocks*.

Voxels. Each block is divided into n *voxels* $V_1, V_2, \ldots, V_n$, which provide the resolution at which precomputed transfer occurs. Note that these voxels are separate from the underlying volume's representation, which can have much higher resolution; i.e., within each precomputed transfer voxel, there can still be many underlying data voxels. We define the *non-empty subset* of voxel V_i as $N_i := \{\mathbf{x} \in \tilde{V}_i \mid \sigma_t(\mathbf{x}, \boldsymbol{\omega}) > 0 \text{ for some } \boldsymbol{\omega}\}$ where $\tilde{V}_i \subset \mathbb{R}^3$ denotes the 3D space contained in voxel V_i.

Interface. We define an *interface* to be the rectangular shared boundary between two neighboring blocks. The final volume can contain millions of distinct interfaces.

Patches. We define *patches* $P_1, P_2, \ldots$ to be oriented faces of voxels on an interface. Oriented-ness means that each patch can be associated with a normal vector $\mathbf{n}(P_i)$, pointing into one of the neighboring blocks of the underlying interface. Patches will serve the purpose of connecting the transfer between blocks.

Figure 4 shows a flatland visualization of a volume consisting of three blocks, each of which is divided into 3×5 voxels. In this example, there are two interfaces and twelve patches. This corresponds directly to our actual algorithm, except the real implementation is in three dimensions, uses millions of blocks, and is optionally warped by a shell-map [Porumbescu et al. 2005] (to bend the volume into a curved shape).

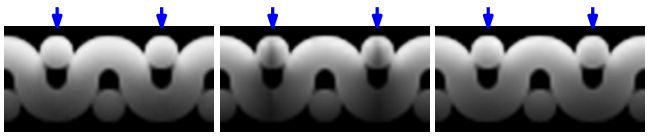


Figure 6: *Cropped 2D slices of the flux field of a synthetic volume with three blocks where the interfaces are indicated with blue arrows: (left) path-traced reference; (center) applying precomputed voxel-to-voxel transfer within blocks leads to darkening, because paths crossing the boundaries are missing; (right) adding transfer across boundaries addresses this energy loss.*

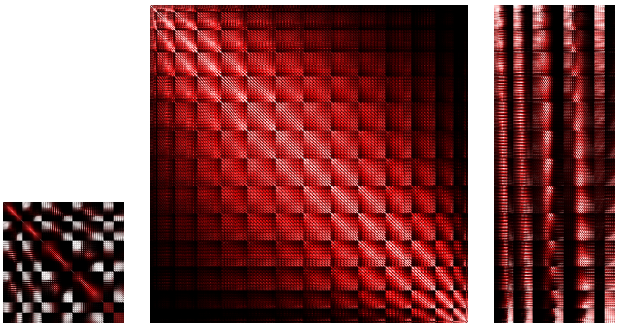


Figure 7: *Visualizations of precomputed patch-to-patch (left), voxel-to-voxel (center), and patch-to-voxel (right) transfer matrices of a twill block with 470 patches and 1239 non-empty voxels.*

3.2 Modular Transfer Pipeline

The pipeline of our system consists of the following stages:

Transfer matrix precomputation. In this stage, we pre-compute the light transfer for a set of exemplar blocks which can later be used to assemble full high-resolution volumes using various techniques including [Zhao et al. 2012]. In particular, we compute three types of transfers, voxel-to-voxel, voxel-to-patch, and patch-to-patch, which will be introduced in Section 4.

Runtime evaluation. At the runtime stage, the pipeline is split into the following three phases (see Figure 5):

1. **Source flux evaluation.** We first evaluate the amount of attenuated light arriving at voxels directly from a light source. This can be treated as the first scattering event, and will serve as the source term for the light propagation of precomputed transfer matrices. Note that, conceptually, the source flux includes the scattering event (more precisely, the material term).
2. **Modular transfer.** Given the source flux, we apply our precomputed transfer to obtain multiple-scattered flux. The voxel-to-voxel transfer captures light transfer within individual blocks, and accounts for the bulk of short range transport. Further, voxel-to-patch and patch-to-patch transfers provide longer-range transport that crosses block boundaries. All of these transfers are introduced in Section 5.2.
3. **Final gathering.** Finally, we run a standard path tracing algorithm accelerated by looking up the previously computed scattered flux field after k scattering events (paths with less than k scatterings are computed by explicit path tracing). More details are available in Section 5.3.

4 Precomputation

In this section, we describe the precomputation stage of our pipeline. Given a block divided into n voxels, we define three kinds of transfer matrices: voxel-to-voxel, voxel-to-patch, and patch-to-patch. Below we give precise definitions of the elements of the transfer matrices, and describe a Monte Carlo particle tracing algorithm to compute them.

Voxel-to-voxel. The *voxel-to-voxel* transfer matrix T^{vv} handles most (but not all) of the light transport: because of the short mean free path in optically dense materials, most transport is local, even though it requires many scattering events. Figure 6 compares the amount of transport contained within the voxel-to-voxel transfer versus the other transfer modes.

The element (i, j) of T^{vv} is defined as a path integral, where the endpoints of the paths are in N_i and N_j, the non-empty subsets of V_i and V_j:

$$T^{vv}(i,j) = \int_{\Omega(N_i,N_j)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{3}$$

Here $\Omega(N_i, N_j)$ means the set of paths (with one or more segments) with endpoints in N_i and N_j, respectively (see the left block in Figure 4). This definition implies that T^{vv} is symmetric. The use of the non-empty subsets N_i instead of the full voxels V_i is important for several reasons. It leads to sparser matrices (and thus less storage). Furthermore, defining the path integral this way helps us to compute total flux within non-empty regions of a voxel, which approximates the radiance values at scattering events better (since these never occur in empty regions).

Algorithm 1 A particle tracing based method for computing the voxel-to-voxel transfer matrix T^{vv} given a block B.

1: $T^{vv} \leftarrow \mathbf{0}$
2: **for** $i = 1$ to numVoxels **do** ▷ Loop over all voxels
3: **for** $t = 1$ to m **do**
4: sample a ray $(\mathbf{x}, \boldsymbol{\omega})$ with $\mathbf{x} \in N_i$ and $\boldsymbol{\omega} \in \mathcal{S}^2$
5: $w \leftarrow 4\pi|N_i|$ ▷ Weight initialization
6: **loop**
7: sample s (with Woodcock tracking) according to

$$p(s) = \sigma_t(\mathbf{x} + s\boldsymbol{\omega}, \boldsymbol{\omega}) \exp\left(-\int_0^s \sigma_t(\mathbf{x} + s'\boldsymbol{\omega}, \boldsymbol{\omega})\,\mathrm{d}s'\right)$$

8: $\mathbf{x}' \leftarrow \mathbf{x} + s\boldsymbol{\omega}$ ▷ Get a scattering event at $\mathbf{x}'$
9: **if** $\mathbf{x}' \in B$ **then**
10: find the voxel j that contains $\mathbf{x}'$
11: **else**
12: **break** ▷ Terminate the path
13: **end if**
14: $T^{vv}(i,j) \leftarrow T^{vv}(i,j) + w/\sigma_t(\mathbf{x}', \boldsymbol{\omega})$ ▷ Deposit energy
15: $w \leftarrow w \cdot \alpha(\mathbf{x}')$ ▷ Update the weight
16: sample direction $\boldsymbol{\omega}'$ according to material term at $\mathbf{x}'$
17: $(\mathbf{x}, \boldsymbol{\omega}) \leftarrow (\mathbf{x}', \boldsymbol{\omega}')$ ▷ Continue the path
18: **end loop**
19: **end for**
20: **end for**
21: $T^{vv} \leftarrow T^{vv}/m$

We compute T^{vv} using Monte Carlo particle tracing, as shown in Algorithm 1. For each voxel V_i, we trace m paths. For each of the m paths we trace the path through the block, with appropriate importance sampling, and deposit values into the appropriate (i, j) entry in the transport matrix T_{vv}. This is done inside the loop on lines 6-18. Each path terminates when it exits the volume (line 12). At each vertex, energy is deposited (line 14) and then multiplied by the albedo (line 15). Note that the deposition is divided by the probability density of the initial particle generation, captured in the initial weight $w = |N_i|$ for position choice (where $|N_i|$ is the volume of the non-empty voxel subset), and 4π for direction choice. The value is also divided by the value of σ_t at the deposition vertex, because it figures in the sampling probability density but not in the path integral we want to compute. Since $\mathbf{x}'$ is chosen by importance-sampling σ_t, it holds that $\sigma_t(\mathbf{x}', \boldsymbol{\omega}) > 0$ in line 14, preventing any division-by-zero issues.

In summary, the particle tracing importance-samples all terms along a path other than the initial probability, albedo terms, and the final division by σ_t, which are exactly the terms that occur explicitly in the weighting.

Voxel-to-patch. As shown in the right block of Figure 4, the *voxel-to-patch* transfer matrix T^{vp} is similarly defined as the integral over

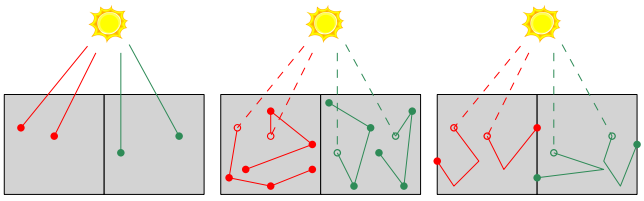


Figure 8: *Light paths captured by (a) source flux $\mathbf{\Phi}^s$; (b) $\mathbf{\Phi}^s$ with voxel-to-voxel transfer applied; (c) $\mathbf{\Phi}^s$ with voxel-to-patch transfer applied.*

paths between a voxel V_i and patch P_j:

$$T^{vp}(i,j) = \int_{\Omega(N_i,P_j)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}), \tag{4}$$

where $\Omega(N_i, P_j)$ is again naturally defined as the set of paths with endpoints in N_i and P_j. Importantly, the non-empty voxel subset N_i is used again. Note the contribution $f(\bar{x})$ includes a cosine term on the path vertex that lies on the patch. This is because geometry terms include a cosine on surfaces, and the patch is treated as a surface. Note that patch-to-voxel transport T^{pv} is the transpose of T^{vp}, so no separate definition is required for it. These matrices are normally not square.

The computation of T^{pv} is handled similarly to T^{vv} by volumetric particle tracing, except the origin of a particle is chosen on a patch, and its direction is chosen by importance-sampling the cosine of the angle from the patch normal.

Patch-to-patch. The *patch-to-patch* transfer matrix T^{pp} is defined as the integral over paths between patches P_i and P_j (see the center block in Figure 4):

$$T^{pp}(i,j) = \int_{\Omega(P_i,P_j)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{5}$$

Computing T^{pp} is analogous to T^{vp}, except deposition occurs at other patches instead of voxels. Matrix T^{pp} is also symmetric.

Summary of precomputation. Assume we have a set of exemplar blocks which can be assembled to construct very large volumes. For the i-th exemplar block, we precompute and store the voxel-to-voxel, voxel-to-patch, and patch-to-patch matrices T_i^{vv}, T_i^{vp}, and T_i^{pp}. Figure 7 visualizes the transfer matrices of a block of twill fabric with shiny red fibers.

5 Runtime Evaluation

In this section, we describe the three phases of the runtime stage of our pipeline: source flux evaluation, modular transfer, and final gathering. The first and third stages are based on standard approaches, while the core complexity lies in the second phase, where the precomputed transfers are applied.

5.1 Source Flux Evaluation

The runtime of our pipeline starts by evaluating the *source flux vector* $\mathbf{\Phi}^s$, which is the amount of single-scattered light arriving at voxels directly from the set of direct light sources (see Figure 8a). Denote the set of emissive surfaces C. Let $\Omega_1(N_i, C)$ be the set of unscattered paths (i.e., direct single-segment connections) from voxel subsets N_i to the set of emissive surfaces C. Denote these single-segment paths by $\bar{x} = \mathbf{x}_0\mathbf{x}_1$. The elements of $\mathbf{\Phi}^s$ can now be defined as:

$$\Phi^s(i) = \int_{\Omega_1(N_i,C)} \int_{S^2} M(\mathbf{x}_0, \boldsymbol{\omega}, \overrightarrow{\mathbf{x}_0\mathbf{x}_1}) f(\bar{x})\,\mathrm{d}\boldsymbol{\omega}\,\mathrm{d}\mu(\bar{x}). \tag{6}$$

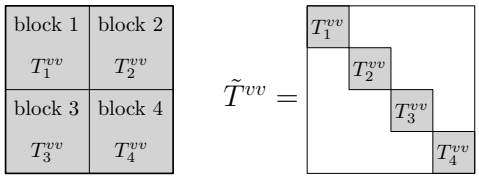


Figure 9: *Formation of block-diagonal matrix $\tilde{T}^{vv}$ for a volume with four blocks.*

An important detail is that, intuitively, we are including in $\mathbf{\Phi}^s$ light that scattered once. More precisely, we include the material term in the definition, integrating it over all directions. This is the first application of an isotropy event: since we do not know $\boldsymbol{\omega}$, the direction light will take after this scattering event, we integrate over all such directions.

We compute $\mathbf{\Phi}^s$ by particle tracing, and store it as a sparse vector where the total number of non-zero entries never exceeds the number of traced particles, and is usually much smaller than the number of voxels. The noise in $\mathbf{\Phi}^s$ is not a problem, because the application of the precomputed transfer matrices multiplies the effective number of paths by a large number.

5.2 Modular Transfer

The goal of the modular transfer phase of the pipeline is to use the precomputed matrices to turn source flux into multiple-scattered flux, which can then be simply looked up by the final gather phase of the pipeline. We define the *ground-truth multiple-scattered flux* $\mathbf{\Phi}^{gt}$ as:

$$\mathbf{\Phi}^{gt}(i) = \int_{\Omega_{2+}(N_i,C)} f(\bar{x})\,\mathrm{d}\mu(\bar{x}). \tag{7}$$

Here, by $\int_{\Omega_{2+}(N_i,C)}$ we mean paths with 2 or more segments. Our goal is to compute an approximate multiple-scattered flux $\mathbf{\Phi}^m$ by application of precomputed transfer, such that $\mathbf{\Phi}^m \approx \mathbf{\Phi}^{gt}$. An important detail is that $\mathbf{\Phi}^m$, unlike $\mathbf{\Phi}^s$, does not conceptually contain the scattering event, i.e. the material term; it will be the responsibility of the final gather phase to include it.

For the final volume (where each block is the copy of an exemplar), let the transfer matrices of the i-th block be T_i^{vv}, T_i^{vp} and T_i^{pp}. We define the *global completions* of these matrices, which operate on a global indexing of all voxels and patches in the volume, and denote them by $\tilde{T}^{vv}$, $\tilde{T}^{vp}$, and $\tilde{T}^{pp}$. If we assign contiguous indices to voxels belonging to the same block and patches belonging to the same interface, then $\tilde{T}^{vv}$ and $\tilde{T}^{pp}$ become block diagonal (see Figure 9). In other words, this is simply an imaginary stacking of the precomputed matrices for different exemplar blocks, so that we have only one large block-diagonal matrix for each transfer type.

Given the source flux vector $\mathbf{\Phi}^s$, the first step of the modular transfer phase is the application of voxel-to-voxel transfer. This is accomplished simply by computing the matrix-vector multiplication $\tilde{T}^{vv}\mathbf{\Phi}^s$ (see Figure 8b).

However, the voxel-to-voxel matrix $\tilde{T}^{vv}$ only encodes transfer over paths that do not cross block interfaces, as shown in Figure 8b. To compensate for this omission, we need to propagate the flux to the block interface, "cross" to the other side of the interface to enter the neighboring block, and then propagate further to voxels in that block.

More precisely, we transfer voxel fluxes into patch fluxes by an application of $\tilde{T}^{vp}$ (see Figure 8c). Now the patch flux on patch P_i describes the amount of light it receives. Let $\mathrm{flip}(i)$ denote the index of the patch that overlaps with i but with the opposite normal direction (so patches P_i and $P_{\mathrm{flip}(i)}$ always belong to immediately neighboring blocks). This flip operator can be written as a permutation matrix Q on the set of global patch indices. To propagate

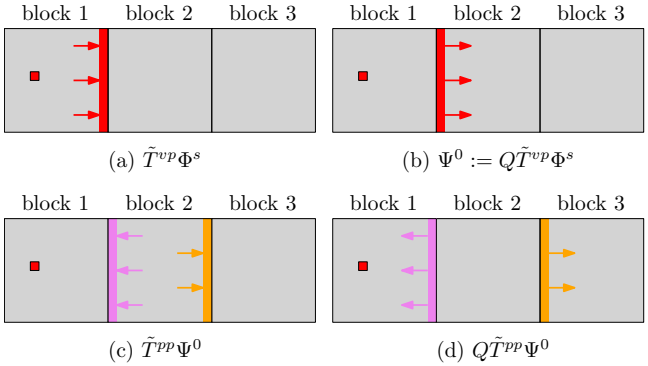


Figure 10: *An example of patch flux propagation. The scene contains 3 blocks and 4 patches defined over 2 interfaces. Assume that all voxels have zero flux except for one in block 1 marked with the red square. Then (a) shows the patch flux received by the right patch in block 1; (b) applying flip operator Q gives patch flux emitted by the left patch in block 2; (c) multiplying by $\tilde{T}^{pp}$ gives the patch flux received by both patches in block 2; (d) applying another Q yields the patch flux emitted by the two patches in blocks 1 and 3.*

the patch flux received by P_i, we make the opposite patch $P_{\text{flip}(i)}$ to emit the same amount of energy into the neighboring block (see Figure 10ab): this is an application of a diffuser event, which prevents the need to store a directionally-dependent function on each patch; instead it suffices to compute a scalar quantity, the patch flux. Note that this assumption is crucial to making our approach tractable since storing directional variation would prohibitively increase the amount of storage. However, this still does not account for all light paths. To allow the energy to go further, we need to apply $Q\tilde{T}^{pp}$ multiple times, to go from patch to patch across entire blocks (see Figure 10cd). This will add the light traversing block boundaries for longer range transport. This eventually leads to the correct expression for computing the multiple-scattered flux:

$$\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}\left(\sum_{a=0}^{\infty}\left(Q\tilde{T}^{pp}\right)^a\right)Q\tilde{T}^{vp}\mathbf{\Phi}^s. \tag{8}$$

Let $U := I - Q\tilde{T}^{pp}$. Rewriting the infinite series in (8) using the Neumann series yields:

$$\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{9}$$

which can be computed as follows. First, we compute by solving a linear system the *patch flux*:

$$\mathbf{\Phi}^p := U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{10}$$

which equals the amount of energy emitted by each patch i contributed by light paths going across one or more interfaces and flipping across the interface. Then we evaluate (9) using $\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \tilde{T}^{pv}\mathbf{\Phi}^p$.

To compute Equation (10), we need to solve a large linear system $U\mathbf{\Phi}^p = Q\tilde{T}^{vp}\mathbf{\Phi}^s$. Since U is usually asymmetric (because of the flip operator Q), conjugate gradient based algorithms cannot be applied here. Instead, we introduce two methods corresponding to finite element and Monte Carlo approaches to solve this system.

Jacobi Iteration. We can simply truncate the Neumann series, which results in Jacobi iteration. We start with $\mathbf{\Psi}^{(0)} = Q\tilde{T}^{vp}\mathbf{\Phi}^s$, and in the t-th iteration, we compute $\mathbf{\Psi}^{(t+1)} = Q\tilde{T}^{pp}\mathbf{\Psi}^{(t)} + \mathbf{\Psi}^{(0)}$. After a few iterations, set $\mathbf{\Phi}^p = \mathbf{\Psi}^{(t)}$. Note that neither $\tilde{T}^{pp}$ nor Q needs to be formed explicitly; instead, for any given vector $\mathbf{v}$, $\tilde{T}^{pp}\mathbf{v}$ can be computed by performing the patch-to-patch transfer on each block, and $Q\mathbf{v}$ can be obtained by permuting $\mathbf{v}$'s elements.

Algorithm 2 Random walk estimating $\mathbf{\Phi}^{ls}(i)$.

```
(j, s) ← sampleRow(T̃^pv, i)        ▷ Jump from voxel to patch
loop
    j ← flip(j)                     ▷ Flip the patch
    Let q be the stop probability
    if rand() < q then              ▷ Russian roulette
        s ← s/(1 − q)
        break
    end if
    (j, v) ← sampleRow(T̃^pp, j)     ▷ Jump to another patch
    s ← sv/q                        ▷ Update the throughput
end loop
(i′, v) ← sampleRow(T̃^vp, j)        ▷ Final jump to voxel
return Φ^s(i′) sv
```

Monte Carlo Matrix Inversion. Although Jacobi iteration works adequately, it may take many iterations to converge and requires storing the full vector $\mathbf{\Psi}^{(t)}$ (whose size equals the total number of patches), which can be very expensive: the tablecloth in Figure 1, for example, contains many millions of block interfaces, each with hundreds of patches. To make our method truly scalable to very large models, we implemented a Monte Carlo method based on [Forsythe and Leibler 1950], which does not require the storage of $\mathbf{\Phi}^p$ explicitly. Let

$$\mathbf{\Phi}^{ls} := \tilde{T}^{pv}\mathbf{\Phi}^p = \tilde{T}^{pv}U^{-1}Q\tilde{T}^{vp}\mathbf{\Phi}^s, \tag{11}$$

which is the component of the multiple-scattered flux that requires solving a linear system (i.e., summing a Neumann series). If elements of $\mathbf{\Phi}^{ls}$ can be efficiently estimated, we can then compute Equation (9) with $\mathbf{\Phi}^m = \tilde{T}^{vv}\mathbf{\Phi}^s + \mathbf{\Phi}^{ls}$.

To do this, we define a *random walk* whose expected value is the i-th element of $\mathbf{\Phi}^{ls}$. The random walk intuitively traces a "path", whose "vertices" are discrete states corresponding to voxel and patch indices, instead of actual scattering points, and its "edges" consist of multi-vertex jumps precomputed within the transfer matrix elements.

The algorithm conceptually starts at voxel V_i, and immediately transitions into the j_1-th patch, with a discrete probability distribution proportional to the i-th row of $\tilde{T}^{pv}$. Then, a series of transitions between patches with indices $j_1, j_2, j_3, \ldots$ is made with probabilities proportional to a corresponding element of the matrix $\tilde{T}^{pp}$. A Russian roulette approach is used to eventually terminate this loop, making a final transition to a voxel index i', with probability based on the corresponding row of $\tilde{T}^{vp}$. At this point, the i'-th element of source flux $\mathbf{\Phi}^s$ is queried and scaled by terms accumulated along the path. It is not difficult to see that the expected value of this process is precisely the matrix expression containing the Neumann series that we are interested in computing. Note that each transition in this random walk can capture many light scattering events which are expensive to simulate exactly.

Algorithm 2 describes the random walk in more detail. In the algorithm, we use a sampleRow(A, i) function, which returns the index of an element sampled from the i-th row of matrix A, with a probability proportional to the element's value. It also returns the value of the element, divided by the probability of choosing it. All the rendered results in Section 6 are generated using this algorithm.

5.3 Final Gathering

Given the discretized flux field inside the volume computed by the modular transfer, we can now render it efficiently. We perform a standard Monte Carlo path tracing algorithm with explicit direct illumination. However, when handling the k-th scattering event on a light path at location $\mathbf{x}$, we replace the phase function by a uniform isotropic one. This is the second time we insert an isotropy event to

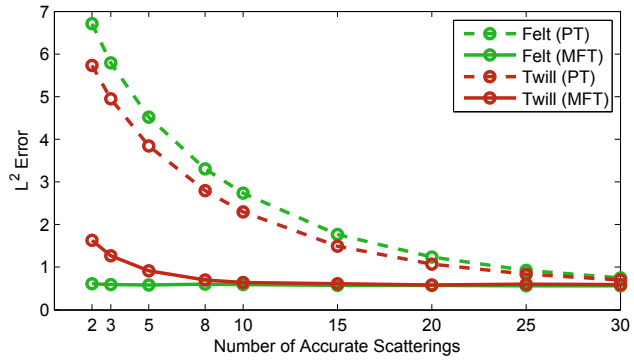


Figure 11: *Convergence experiment: we rendered multiple results with our method (solid lines) and path tracing with terminating the path after k scatterings (dashed lines) using varying k values and computed their $\mathcal{L}^2$ error (plotted as $\log(1+y)$ for error y). Note that the graphs do not converge to zero, because there is Monte Carlo noise in both images being compared.*

obtain a separable approximation to the path integral. This lets us stop the recursive process and approximate the indirect illumination at point $\mathbf{x}$ as $\mathbf{\Phi}^m(i)/(4\pi|N_i|)$ where $\mathbf{\Phi}^m(i)$ is the stored multiple-scattered flux value computed by the modular transfer. The final gather computes all paths with less than k scattering events using standard path tracing.

Finally, we choose the value of k as follows. As shown in Figure 11, for a range of materials, we produced multiple renderings with changing k values. The results indicate that for materials with random structures, such as felt, a k value of 2 or 3 is sufficient. For structured materials with parallel shiny fibers (such as the twill), a value of $k = 6$ produces high quality results, and our approach still converges much faster than path tracing. Therefore, we picked $k = 6$ for all our results.

6 Results

In this section, we first demonstrate the performance of our technique by comparing flux fields computed by our method, path tracing, and photon mapping. Afterwards we show renderings for a range of materials created with our method.

6.1 Flux Field Visualizations

Figure 12 shows the multiple-scattered flux fields obtained by path tracing and our technique. The entire volume contains over one million blocks, and in this figure we show 2D slices across 20 of them. The top of the figure shows the reference Φ_{gt} generated by tracing thousands of paths for each non-empty voxel, while the bottom shows the approximate Φ_m generated by tracing 50 million particles for the source flux and solving the modular transfer described in Section 5.2. Our result matches the ground truth well.

6.2 Photon Mapping Comparisons

Figure 13 shows detail renderings generated by path tracing (the ground truth), our technique (MFT), and photon mapping. Here we

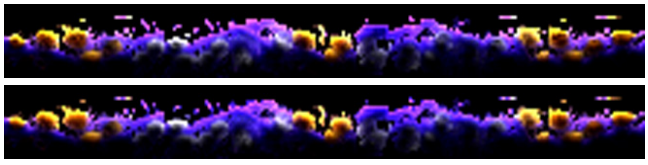


Figure 12: *2D slices of flux fields in non-empty voxels computed with path-tracing (top) and MFT (bottom).*

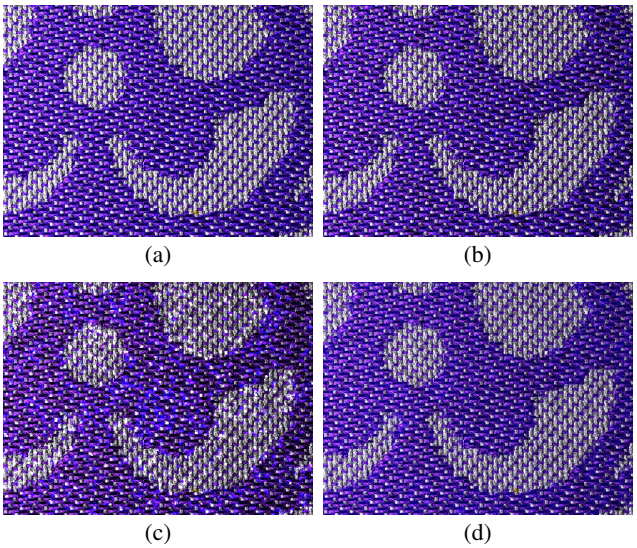


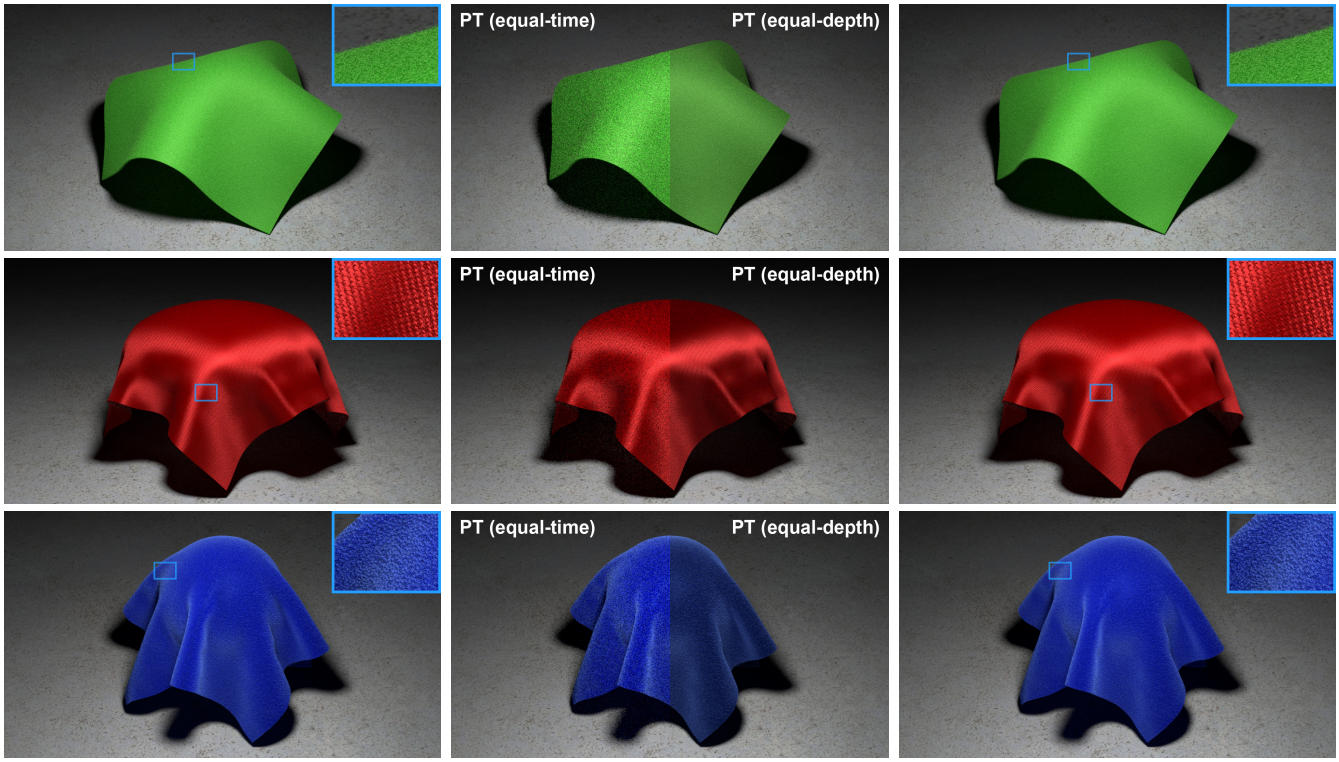
Figure 13: *Images rendered with (a) standard path tracing in 1.3 hours; (b) MFT in 12 minutes (with 10M particles traced); (c) volume photon mapping in 20 minutes (with 100M photons stored); (d) volume photon mapping in 1 hour (with a billion photons stored).*

use $k = 3$, i.e. three path-traced events precede the MFT or photon map lookup. The entire scene is a single sheet of cloth with over one million blocks, of which a small patch (containing 0.24% of all traced particles) is shown. Figure 13b is obtained by tracing 10 million particles for the source flux. Our method is much faster than path tracing while providing good accuracy. Figure 13c is generated with photon mapping with 100 million particles. Artifacts result from the stored photon density being insufficient to capture the yarn-level structures. In Figure 13d, we show a photon mapping result using one billion photons.

6.3 Rendered Results

Next, we show renderings of a variety of volumetric appearance models of fabrics created with techniques introduced by Zhao et al. [2011; 2012]. Such models often consist of trillions of micron-resolution (effective) voxels, making them very challenging to render. In our experiments, we focus on demonstrating our technique over such highly complex volumetric models. We used a simple lighting configuration and compared our results with those created by volume path tracing. We implemented our system based on the Mitsuba physically based renderer [Jakob 2010], and ran all our experiments on an Intel server equipped with four Xeon X7560 eight-core CPUs.

Precomputation. When precomputing the transfer matrices required by MFT, for each material, we pick a resolution such that every exemplar block has around 1000 non-empty voxels and 500 patches. As previously mentioned, the choice of this resolution does not depend on the actual data resolution of the underlying volume. In our case, each cloth block (representing one yarn crossing) contains about one million data voxels, and the final volumes in Figures 14 and 15 contain 2.5×10^{11} and 1.4×10^{12} effective voxels, respectively. Precomputation of a block takes roughly 4 hours, and storing the transfer matrices (as OpenEXR images) takes about 20MB. We distributed the precomputation tasks to Amazon Elastic Compute Cloud (EC2) by using between 25 and 80 `c1.xlarge` instances, where processing each exemplar block costs 2 USD. Note that the matrices are purely determined by material properties. Thus, after a one-time precomputation, we can use MFT to accelerate renderings of the material under any lighting condition. Also, we use the same precomputation, which was performed over rectangular blocks, when the volume is warped into draped shapes with shell-mapping (Figures 1, 14 and 15). Furthermore, one set of



MFT path tracing path-traced reference

Figure 14: *Rendered fabrics in draped configurations: (top) felt; (middle) twill weave; (bottom) velvet. The left column shows results rendered by our method (MFT). The center column shows two path-traced results: the left half of each image is rendered with fewer paths, sampled to achieve similar rendering time, but therefore, exhibits higher noise; the right half is rendered with the same number of samples but terminating all paths after 6 scatterings, showing significant darkening because of the lack of high-order scatterings. The right column shows path-traced references computed in much longer time. See Table 2 for performance numbers.*

Scene	Total Blocks	Exemplar Blocks	Precomp. Time	Path Length		Time Cost		Speed Up
				PT	MFT	PT	MFT	
Felt	250 000	25	100	44.1	7.4	144	14	10.3×
Twill	250 000	25	100	37.1	7.2	90	10	9.0×
Velvet	250 000	25	100	71.0	7.9	192	15	12.8×
Damask	1 350 000	120	480	65.8	7.2	108	15	7.2×
Wood	256	25	100	17.9	6.3	17	5.6	3.0×
Synthetic	625	25	100	23.6	6.9	6.6	1.6	4.1×

Table 2: *Scene statistics. The table shows the number of blocks in the scene, the number of exemplar blocks, the precomputation time for all exemplars (in hours), average path length, and rendering time (in minutes) for path tracing (PT) and our method (MFT). Felt, twill, and velvet correspond to Figure 14; damask to both designs in Figure 15; wood and synthetic to Figure 16. The MFT rendering time includes the portion spent on computing the source flux $\mathbf{\Phi}^s$ (by tracing particles from light sources), which is less than 2 minutes for all our results. The Monte Carlo matrix inversion step takes less than 15% of the rendering time for all scenes.*

exemplar blocks may be used to synthesize models with very different structures (such as the cloth with different weave patterns in Figure 1 and both rows of Figure 15), amortizing the precomputation cost over a very large set of designs.

Single-Colored Fabrics. We first show rendered results for three types of fabrics (Figure 14) where the single-scattering albedo for each material stays constant (namely all fibers have identical colors) and equals around 0.975 (for the brightest color channel). Our method achieved respectively $10.3\times$, $9.0\times$, and $12.8\times$ speedup for these materials. The top row contains renderings for felt, a thick non-woven fabric with layers of disorganized fibers. The middle row shows those for a twill fabric, conveying characteristic diagonal lines. Unlike felt, this fabric contains well aligned fibers in two perpendicular directions which create strong anisotropic highlights. The bottom row exhibits rendered images of velvet with a visible surface composed of fibers sticking up from the base material, creating a distinctive grazing-angle highlight. Each of the three results has 25 precomputed exemplar blocks which can be reused to synthesize the same kind of material in arbitrary sizes. This same data was also used to produce all the turn-table and zoom sequences in the accompanying video.

Fabrics with Complex Designs. In addition, we created an exemplar database with 120 blocks using the method from [Zhao et al. 2012]. With such a database, fabrics with complicated weave patterns can be constructed through synthesis. Since we precompute the transfer matrices for each exemplar block, the precomputation can be reused to render any synthesized cloth. Figures 1, 15 show renderings with three designs generated with this single database where our approach speeds up the renderings by $7.2\times$. A high-resolution version of the left image in Figure 1 is available as supplementary material, and animated renderings with a moving light source are included in the video.

More information for images in Figures 14, 15 and 16 are available in Table 2. The results show that path tracing suffers from very long light paths because of the volume complexity and high albedo values. On the other hand, using the MFT method, we terminate the path after 6 scatterings, which bounds the maximal path length and yields significant speedups.

Materials Beyond Cloth. Finally, our technique can also be applied to non-cloth materials, as long as they are formed using a small set of exemplar blocks. Figure 16 shows two such results. On the left, we show a piece of finished wood represented with a micro-

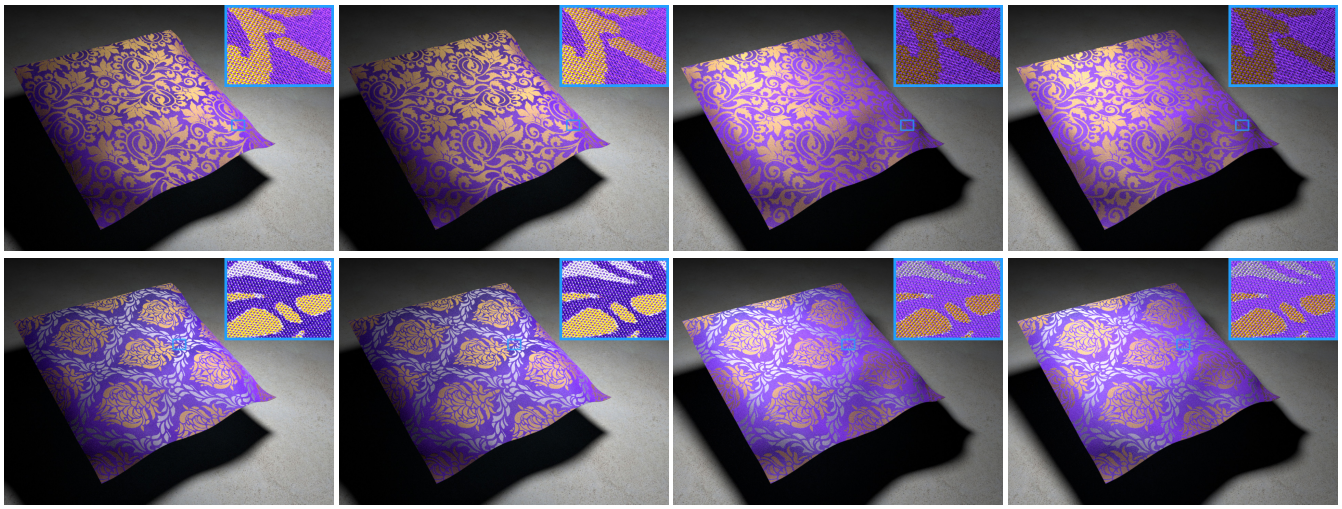


Figure 15: *Fabrics with two designs (both with* 900×1500 *blocks) rendered under two lighting configurations. All results rendered with our technique use the same set of precomputed transfer matrices. Performance information is in Table 2, and more designs are available as supplementary materials.*

flake volume based on the data from [Marschner et al. 2005]. The rendered wood conveys characteristic anisotropic highlights. Our method captures all these effects while filling in high-order scatterings accurately. On the right, we show a synthetic volume with a fine, coral-like structure made of a solid texture provided by [Kopf et al. 2007]. Our result matches the ground truth very well. Although these datasets contain only a few million effective voxels, and are much less complicated than the cloth volumes, our method still achieved $3 - 4.1\times$ speedup over path tracing.

7 Conclusion and Future Work

In this paper, we introduced a precomputation-based approach to accelerate renderings of very complex volumetric materials built from sets of exemplar blocks. These materials are slow to render using pure Monte Carlo techniques and do not easily allow for diffusion approximations. Our algorithm separates low-order and high-order scattering and approximates the latter using a modular flux transfer framework. Based on the observation that introducing diffuser and isotropy events to long-enough light paths has little effect on accuracy, we showed that those paths can be split into precomputed and runtime components. The former can be evaluated by precomputing voxel-to-voxel, voxel-to-patch, and patch-to-patch transfers for each exemplar block. The precomputation cost is amortized over many different lightings, shapes, and designs. An important component of our solution is a Monte Carlo matrix inversion method to solve the transfer problem with minimal storage cost; this means that our method has similar scalability to path tracing, but effectively traces much shorter paths for the same quality. Our results demonstrate a speed-up of more than an order of magnitude for thick cloths. In addition, the method can be used for non-cloth materials. We believe our algorithm could be directly used for high-quality rendering in interior design or movie production.

One limitation of our current work is that we require a new precomputation if the exemplar blocks change their optical properties, such as single-scattering albedo. In the future, we plan to extend our framework to permit precomputation with material editing. Our Monte Carlo particle tracing based precomputation stage is quite expensive, thus further optimizations would be very useful. We would also like to explore heuristics for choosing the value of k adaptively, such as ones based on a frequency analysis of the resulting error. Combining our approach with lightcuts [Walter et al. 2012] or other algorithms may be an interesting direction. Finally, we would like to push our approach to an even larger scale, for example, rendering a crowd of clothed characters.

Acknowledgements

Funding for this work was provided by NSF IIS grants 1011832, 1115242, 1161645 and 1011919, ONR PECASE grant N00014-09-1-0741, the Intel Science and Technology Center for Visual Computing, and equipment and funding from NVIDIA, Adobe and Pixar.

References

ARBREE, A., WALTER, B., AND BALA, K. 2011. Heterogeneous subsurface scattering using the finite element method. *IEEE Transactions on Visualization and Computer Graphics 17*, 7, 956–969.

ARNALDI, B., PUEYO, X., AND VILAPLANA, J. 1994. On the division of environments by virtual walls for radiosity computation. In *Photorealistic Rendering in Computer Graphics*. Springer, 198–205.

BEKAERT, P. 1999. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, Katholieke Universiteit Leuven.

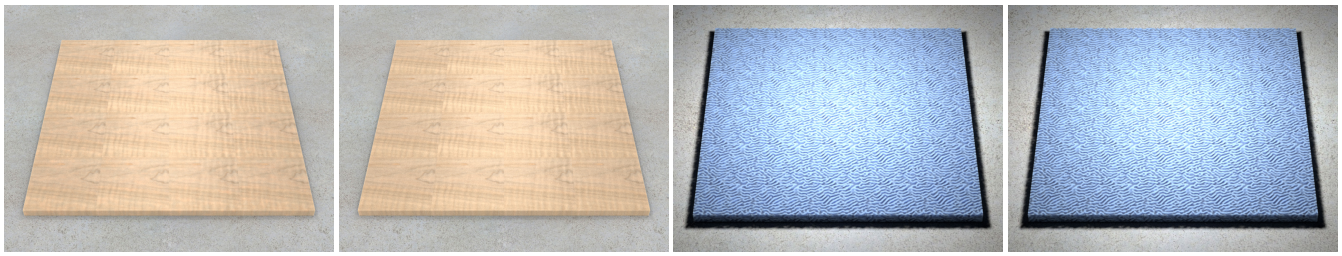
D'EON, E., AND IRVING, G. 2011. A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph. 30*, 4, 56:1–56:14.

DONNER, C., AND JENSEN, H. W. 2005. Light diffusion in multi-layered translucent materials. *ACM Trans. Graph. 24*, 3, 1032–1039.

DONNER, C., AND JENSEN, H. W. 2007. Rendering translucent materials using photon diffusion. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, Eurographics Association, 243–251.

FATTAL, R. 2009. Participating media illumination using light propagation maps. *ACM Trans. Graph. 28*, 1, 7:1–7:11.

FORSYTHE, G., AND LEIBLER, R. 1950. Matrix inversion by a monte carlo method. *Mathematical Tables and Other Aids to Computation*, 127–129.



MFT path-traced reference MFT path-traced reference

Figure 16: *Renderings of materials beyond cloth under different lightings: (left) finished wood; (right) synthetic volume. Please see Table 2 for more information.*

JAKOB, W., ARBREE, A., MOON, J. T., BALA, K., AND MARSCHNER, S. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph. 29*, 4, 53:1–53:13.

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

JAROSZ, W., NOWROUZEZAHRAI, D., SADEGHI, I., AND JENSEN, H. W. 2011. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph. 30*, 1, 5:1–5:19.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 311–320.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 511–518.

KAJIYA, J. T., AND VON HERZEN, B. P. 1984. Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 165–174.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2D exemplars. *ACM Trans. Graph. 26*, 3, 2:1–2:9.

LENSCH, H., GOESELE, M., BEKAERT, P., KAUTZ, J., MAGNOR, M., LANG, J., AND SEIDEL, H. 2003. Interactive rendering of translucent objects. In *Computer Graphics Forum*, vol. 22, 195–205.

LEWIS, R. R., AND FOURNIER, A. 1996. Light-driven global illumination with a wavelet representation of light transport. In *In Seventh Eurographics Workshop on Rendering*, Springer, 11–20.

LOOS, B. J., ANTANI, L., MITCHELL, K., NOWROUZEZAHRAI, D., JAROSZ, W., AND SLOAN, P.-P. 2011. Modular radiance transfer. *ACM Trans. Graph. 30*, 6, 178:1–178:10.

MARSCHNER, S. R., WESTIN, S. H., ARBREE, A., AND MOON, J. T. 2005. Measuring and modeling the appearance of finished wood. *ACM Trans. Graph. 24*, 3, 727–734.

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

MOON, J. T., WALTER, B., AND MARSCHNER, S. R. 2007. Rendering discrete random media using precomputed scattering solutions. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, Eurographics Association, 231–242.

MOON, J. T., WALTER, B., AND MARSCHNER, S. 2008. Efficient multiple scattering in hair using spherical harmonics. *ACM Trans. Graph. 27*, 3, 31:1–31:7.

NARASIMHAN, S. G., AND NAYAR, S. K. 2003. Shedding light on the weather. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, IEEE Computer Society, 665–672.

PAULY, M., KOLLIG, T., AND KELLER, A. 2000. Metropolis light transport for participating media. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, 11–22.

PORUMBESCU, S. D., BUDGE, B., FENG, L., AND JOY, K. I. 2005. Shell maps. *ACM Trans. Graph. 24*, 3, 626–633.

PREMOŽE, S., ASHIKHMIN, M., TESSENDORF, J., RAMAMOORTHI, R., AND NAYAR, S. 2004. Practical rendering of multiple scattering effects in participating media. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, Eurographics Association, 363–374.

RUSHMEIER, H. E., AND TORRANCE, K. E. 1987. The zonal method for calculating light intensities in the presence of a participating medium. *SIGGRAPH Comput. Graph. 21*, 4, 293–302.

SCHRODER, K., KLEIN, R., AND ZINKE, A. 2011. A volumetric approach to predictive rendering of fabrics. *Comput. Graph. Forum 30*, 4, 1277–1286.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

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

WANG, J., ZHAO, S., TONG, X., LIN, S., LIN, Z., DONG, Y., GUO, B., AND SHUM, H.-Y. 2008. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph. 27*, 1, 1–18.

XU, H., PENG, Q.-S., AND LIANG, Y.-D. 1990. Accelerated radiosity method for complex environments. *Computers & Graphics 14*, 1, 65–71.

XU, Y.-Q., CHEN, Y., LIN, S., ZHONG, H., WU, E., GUO, B., AND SHUM, H.-Y. 2001. Photorealistic rendering of knitwear using the lumislice. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 391–398.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph. 30*, 4, 44:1–44:10.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2012. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph. 31*, 4, 75:1–75:10.

ZINKE, A., YUKSEL, C., WEBER, A., AND KEYSER, J. 2008. Dual scattering approximation for fast multiple scattering in hair. *ACM Trans. Graph. 27*, 3, 32:1–32:10.