Eurographics Symposium on Rendering (DL-only Track) (2020)
C. Dachsbacher and M. Pharr (Editors)

Multi-Scale Appearance Modeling of Granular Materials with Continuously Varying Grain Properties

Cheng Zhang[1], Shuang Zhao[1]

[1]University of California, Irvine

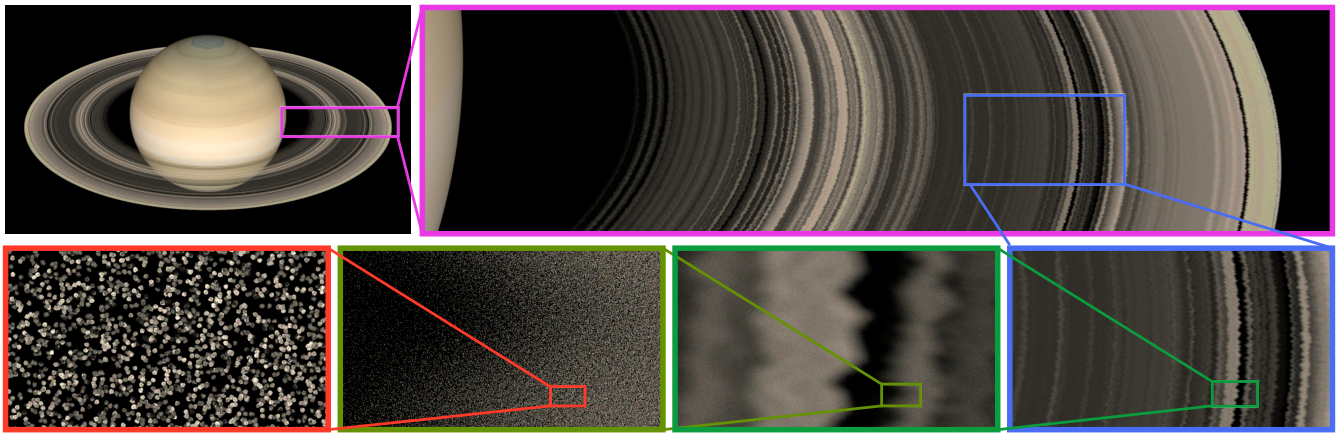


Figure 1: We introduce a new technique for **multi-scale appearance modeling of granular materials** with *complex and spatially varying grain distributions and optical properties*. This scene is modeled after the planet Saturn with its iconic ring depicted using 100 billion grains. Our technique allows this very large virtual scene to be fully described with 3.4 GB of data (including all precomputed information for efficient multi-scale rendering).

Abstract

Many real-world materials such as sand, snow, salt, and rice are comprised of large collections of grains. Previously, multi-scale rendering of granular materials requires precomputing light transport per grain and has difficulty in handling materials with continuously varying grain properties. Further, existing methods usually describe granular materials by explicitly storing individual grains, which becomes hugely data-intensive to describe large objects, or replicating small blocks of grains, which lacks the flexibility to describe materials with grains distributed in nonuniform manners.

We introduce a new method to render granular materials with continuously varying grain optical properties efficiently. This is achieved using a novel symbolic and differentiable simulation of light transport during precomputation. Additionally, we introduce a new representation to depict large-scale granular materials with complex grain distributions. After constructing a template tile as preprocessing, we adapt it at render time to generate large quantities of grains with user-specified distributions. We demonstrate the effectiveness of our techniques using a few examples with a variety of grain properties and distributions.

CCS Concepts

• ***Computing methodologies*** → ***Rendering; Ray tracing;***

1. Introduction

Materials comprised of *discrete grains*, usually terms as *granular materials*, are ubiquitous in the real world. A pile of sand, for instance, contains hundreds of thousands of mineral grains; a snowman is comprised of numerous ice flakes; A granular material's macro-scale appearance is mostly determined by the shape, optical properties, and spatial distributions of the constituent grains.

Traditionally, granular materials are usually described using only

statistical properties such as spatial densities of the grains. Although these compact representations have produced plausible appearance when viewed at distances where individual primitives are unresolvable, they lack the details needed for producing realistic closeup appearances. Further, these models are incapable of capturing visually important phenomena due to the discrete nature of grains such as specular highlights on individual grains).

Recently, several techniques have been developed to efficiently model and render granular materials [MPH*15, MPG*16]. By providing explicit expressions of individual grains, these methods have led to renderings of granular materials with remarkable fidelity and details. To achieve efficient rendering at both the macro and the micro scales, these methods precompute light transport for each type of grain with fixed shapes and optical properties. Although this works adequately for materials containing instances of a small number of example grains, the required precomputation becomes impractical for materials with smoothly changing grain properties. Further, to model large-scale granular materials, existing methods replicate a small 3D tile of grains [MPH*15]. This tiling-based approach works well for materials that are largely homogeneous but lacks the generality to capture more complex grain distributions.

We introduce a new technique to efficiently model the appearance of granular materials at multiple scales. Compared to previous methods, our technique is more general and supports granular materials with: (i) continuously varying grain optical properties; and (ii) large quantities of grains distributed in complex and nonuniform ways. This improved generality can enable the rendering of complex granular materials with richly diverse appearances (see Figures 1, 9–12).

The main contributions of this paper include:

- A symbolic and differentiable Monte Carlo process that allows a single precomputation to be reused for grains with varying optical properties;
- A new method to importance sample the precomputed light transport information at render time;
- A new tile-based representation capable of describing large-scale granular materials with non-uniform grain distributions.

We validate the accuracy of our method by comparing rendered results generated using explicit path tracing as well as previous methods. We further demonstrate the effectiveness of our approach using a few examples.

2. Related Work

Micro-appearance models. A material's fine-grained details can greatly impact how the material appears at much greater scales. Consequently, several material models, usually referred to as *micro-appearance models*, have been developed (e.g., [ZJMB11, DWMG15]). These models explicitly express materials' micro-geometries and offer remarkable fidelity and details. Our model broadly belongs to this category as it offers grain-level details.

Procedural material models. A few previous methods utilize specialized procedures to generate micro-geometries for complex materials such as cloth [SZK15, ZLB16] and wood [LDHM16]. Some of these techniques (e.g., [LZB17]) allow the micro-geometries to be realized at render time. Unfortunately, all these techniques are material-specific, and none of them applies to granular materials, the main focus of this paper.

Discrete stochastic microfacet models. Jakob et al. [JHY*14] introduced a technique that leverages recursive stochastic processes to describe normal variations of a surface's micro-geometry. The key idea is to recursively subdivide collections of microfacets, allowing individual facets to be resolved on the fly in an efficient and consistent way. How we handle large-scale models shares a similar flavor as this technique: instead of storing individual grains, we resolve them on the fly.

Granular material rendering. Efficient rendering of granular materials such as sand, salt, and snow has been studied by a few prior works. Moon et al. [MWM07] introduced a precomputation-based method to render granular materials. By precomputing light transport in groups of grains, a shell tracing algorithm can then use the precomputed information to speed up the rendering process dramatically. Although this technique lacks the flexibility for handling large-scale heterogeneous materials, it had motivated future works on this topic and was adopted by some of them (e.g, [MPG*16]).

Recently, a few techniques have been proposed to model and render granular materials at multiple physical scales. Meng et al. [MPH*15] introduced a framework to model and render granular materials with densely packed grains. At the finest scale, individual grains are explicitly expressed as a translucent object with provided surface geometry as well as interfacial and volumetric scattering profiles. At the macro scale, a granular material is treated as a homogeneous medium and can be rendered using volumetric path tracing (VPT) and diffusion approximation (DA). By switching between these representations at render time, this framework enables multi-scale rendering of granular materials with a good balance between performance and accuracy.

Müller et al. [MPG*16] later adapted this framework to render granular materials with heterogeneity. At the micro scale, this work introduces a new representation that approximates individual grains as simple spheres with light scattering properties (i.e., grain scattering distribution functions or GSDFs) precomputed from their original representations. Additionally, shell tracing was used (instead of DA) at the macro scale for better physical accuracy.

The development of both techniques has brought the visual fidelity of rendered granular materials to a new level. Unfortunately, two fundamental limitations remain. First, both methods require precomputing light transport information per grain (or grain type), which scales poorly to materials with richly diverse or continuous changing grain properties. Second, they have difficulties in modeling large-scale materials with complex nonuniform grain distributions.

Our technique addresses these limitations by significantly generalizing the multi-scale framework. Our method can handle grains with continuous changing properties with minimal precomputation. Additionally, our approach is capable of efficiently describing large-scale granular materials with massive numbers of grains distributed in complex ways. Thanks to these generalizations, our

technique enables the rendering of large-scale granular materials with high heterogeneity, leading to visual effects that cannot be produced with the existing methods.

Symbolic and differentiable rendering. Our work is also closely related to previous Monte Carlo methods that are symbolic or differentiable. Hašan and Ramamoorthi [HR13] developed a technique that enables interactive editing of single-scattering albedo by precomputing pixel intensities as polynomials of albedo. Our method also uses polynomials of grin albedo but in a very different context involving importance sampling of polynomial-valued functions.

Several prior work differentiated Monte Carlo light transport for solving inverse rendering problems [GZB*13, KSZ*15, GLZ16, ZWDR16, LADL18]. These methods are not symbolic, and the estimated gradients are generally used to drive stochastic gradient descent (SGD) optimizations. Our method estimates derivatives of polynomial coefficients with respect to grain optical densities, which is quite different from all these previous methods.

Aggregate scattering. Previously, Blumer et al. [BNH*16] introduced an approach to precompute and store aggregate scattering operators (AGOs) that describe the overall scattering behavior of virtual assets. Our technique shares a similar flavor but differs in two major ways. First, our technique allows a single precomputation to be reused for grains with varying optical properties. Second, our method is specialized to granular materials and utilizes a multi-scale framework that is quite different from the AGOs.

Non-exponential media. Our method is also conceptually related to prior works studying how light scatters within participating media comprised of micro-scale particles distributed in correlated manners [BRM*18, JAG18, GCH*19]. These techniques consider only the statistical properties of the particle distributions, and the particles are normally assumed to be much smaller than individual grains.

3. Multi-Scale Appearance Modeling of Grains with Continuously Varying Properties

We aim to model the appearance of heterogeneous granular materials in which the constituent grains vary in optical properties and are distributed spatially in nonuniform ways.

To this end, we leverage multi-scale techniques introduced by prior works [MPH*15, MPG*16], which we breifly review in 3.1, and introduce a novel technique to render grains with continuously varying optical properties (i.e., optical density and albedo) while offering a good balance between accuracy and performance (§3.2–§3.4).

Additionally, to model large-scale scenes with massive numbers of grains, we present in §4 a new tile-based method to avoid explicitly storing individual grains. At the core of this method is a template tile that can be realized on the fly to form individual tiles with varying grain counts and radii.

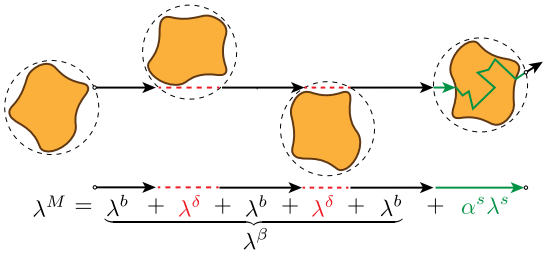


Figure 2: The **mean-free path** λ^{M} of a granular material's medium approximation captures the expected distance light needs to travel after leaving a grain for hitting and exiting a next one.

3.1. Preliminaries

We now briefly review the key concepts, introduced by [MPH*15, MPG*16], for multi-scale renderingof granular materials upon which our technique is built. The basic idea is to use a series of approximated representations of the original model so that one with proper amount of details can be used on the fly to offer a good balance between performance and accuracy.

3.1.1. Grain-Based Modeling

To accelerate light transport simulations at the grain level, a special type of BSSRDF termed as the *grain scattering distribution function (GSDF)* has been introduced by Müller et al. [MPG*16]. Specifically, a GSDF $f_{\mathrm{g}}(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}})$ describes the amount of light leaving the bounding sphere of a grain at $\boldsymbol{x}_{\mathrm{o}}$ with direction $\boldsymbol{\omega}_{\mathrm{o}}$ when entering from $\boldsymbol{x}_{\mathrm{i}}$ with direction $\boldsymbol{\omega}_{\mathrm{i}}$. To reduce the GSDF's dimensionality and improve its practical usefulness, the grains are typically assumed to be oriented randomly, allowing the omission of $\boldsymbol{x}_{\mathrm{o}}$ and the azimuthal component of $\boldsymbol{\omega}_{\mathrm{o}}$, yielding

$$f_{\mathrm{g}}(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}}) \approx \alpha_{\mathrm{g}}^{0}(\beta_{\mathrm{o}})\, \delta(\boldsymbol{\omega}_{\mathrm{i}} + \boldsymbol{\omega}_{\mathrm{o}}) + \alpha_{\mathrm{g}}^{+}(\beta_{\mathrm{o}})\, p_{\mathrm{g}}(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \beta_{\mathrm{o}}), \tag{1}$$

where β_{o} denotes the angle between $\boldsymbol{\omega}_{\mathrm{o}}$ and the surface normal (of the bounding sphere) at $\boldsymbol{x}_{\mathrm{o}}$. Additionally, α_{g}^{0}, α_{g}^{+} capture the probabilities for a light ray with incident angle β_{o} to miss the actual grain and to leave after interacting with it, respectively. Note that α_{g}^{0} and α_{g}^{+} can sum to less than one due to light being absorbed by the grain. When a light ray misses, its direction remains unchanged, which is captured by the Dirac delta function δ in Eq. (1). When the ray hits the grain, a 5D function p_{g} is used to capture the joint distribution of $\boldsymbol{x}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{i}}$ given β_{o}. This function can be further approximated in separate form as

$$p_{\mathrm{g}}(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \beta_{\mathrm{o}}) \approx p_{\mathrm{g}}^{\boldsymbol{x}}(\boldsymbol{x}_{\mathrm{i}}, \beta_{\mathrm{o}})\, p_{\mathrm{g}}^{\boldsymbol{\omega}}(\boldsymbol{\omega}_{\mathrm{i}}, \beta_{\mathrm{o}}). \tag{2}$$

After the GSDF f_{g} is obtained for each grain, it can then be used to accelerate the rendering by avoiding the simulation of individual subsurface scatterings within the grains.

3.1.2. Medium-Based Modeling

When the grain-level details are unnecessary or invisible, granular materials can be approximated as continuous participating media whose optical properties are depicted with (potentially spatially varying) *optical density* σ^{M} and *scaled phase function* f^{M} given

by the normalized phase function f^{M} multiplied by the single-scattering albedo a^{M} (i.e., $\hat{f}^{\mathrm{M}} = a^{\mathrm{M}} f^{\mathrm{M}}$).

Meng et al. [MPH*15] have demonstrated that σ^{M} and $\hat{f}^{\mathrm{M}}$ can be determined based on the grains' optical properties and spatial distributions as follows. The optical density σ^{M} is given by the reciprocal of the *mean-free path* λ^{M}:

$$\sigma^{\mathrm{M}} = 1/\lambda^{\mathrm{M}}, \tag{3}$$

where

$$\lambda^{\mathrm{M}} = \lambda^{\beta} + \alpha^{\mathrm{s}} \lambda^{\mathrm{s}}, \tag{4}$$

with λ^{β} indicating the expected distance for a ray to travel before hitting a next bounding sphere; α^{s} denoting the fraction of rays that hits the grain and eventually leaving its bounding sphere (among all rays hitting the bounding sphere); and λ^{s} being the mean offset of a ray when it interacts with a grain in its bounding sphere. Figure 2 summarizes the definitions of these quantities.

In Eq. (4), λ^{β} is in turn given by

$$\lambda^{\beta} = \left(\lambda^{\mathrm{b}} + \lambda^{\delta}\right) \frac{1-\beta}{\beta} + \lambda^{\mathrm{b}}, \tag{5}$$

where λ^{b} is the expected distance to the next bounding sphere; λ^{δ} is the mean distance traveled by the ray when it misses the grain; and β is the conditional probability for the ray to hit the grain given it hits the bounding sphere. In Eq. (5), the parameters λ^{b}, λ^{β} are obtained via

$$\lambda^{\mathrm{b}} = \frac{4r^{(3)}}{3r^{(2)}} \frac{1-f}{f}, \qquad \lambda^{\delta} = \frac{1}{N} \sum_{n=1}^{N} \hat{\lambda}_n^{\delta}, \tag{6}$$

where f is the *packing rate* [Dul12] that captures the fraction of volume filled by the grains, N is the number of grains, and $r^{(2)}$, $r^{(3)}$ respectively indicate the average square and cubic radii of all grains. Additionally, $\hat{\lambda}_n^{\delta}$ denotes the expected ray offset when it misses the n-th grain (but hits its bounding sphere).

The remaining terms α^{s} and λ^{s} in Eq. (4) capture how light interacts with the grains (via reflection, refraction, and subsurface scattering) and are respectively given by

$$\alpha^{\mathrm{s}} = \frac{1}{N} \sum_{n=1}^{N} \hat{\alpha}_n^{\mathrm{s}}(\sigma_n, a_n), \qquad \lambda^{\mathrm{s}} = \frac{1}{N} \sum_{n=1}^{N} \hat{\lambda}_n^{\mathrm{s}}(\sigma_n, a_n), \tag{7}$$

where $\hat{\alpha}_n^{\mathrm{s}}$ and $\hat{\lambda}_n^{\mathrm{s}}$ are the effective albedo given by Eq. (9) and the mean ray offset (when it intersects with the actual grain) of the n-th grain, respectively.

Besides σ^{M}, the scaled phase function $\hat{f}^{\mathrm{M}}$ is given by

$$\hat{f}^{\mathrm{M}}(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o) = \frac{1}{N} \sum_{n=1}^{N} \hat{f}_n(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{o}} \,|\, \sigma_n, a_n), \tag{8}$$

where $\hat{f}_n$ is the effective scaled phase function of the n-th grain which also determines the grain's effective albedo:

$$\hat{\alpha}_n^{\mathrm{s}}(\sigma_n, a_n) = 2\pi \int_{-1}^{1} \hat{f}_n(x \,|\, \sigma_n, a_n)\, \mathrm{d}x. \tag{9}$$

To allow medium-level heterogeneity (i.e., spatially varying σ^{M} and $\hat{f}^{\mathrm{M}}$), we store the material properties using 3D grids and evaluating Eqs. (3–8) in a per-voxel fashion.

The rest of this section (§3.2–§3.4) provides a detailed description of our multi-scale modeling technique.

3.2. Grain-Based Modeling

We model the shapes of individual grains with polygonal meshes and the materials as homogeneous participating media (with optional refractive boundaries). The brute-force way to render a granular material at the grain level is to use explicit path tracing (EPT).

Unfortunately, EPT is highly inefficient due to high model complexity as well as the need to simulate multiple scattering inside grains. To address this problem, we leverage GSDFs (§3.1.1) to efficiently model light-grain interactions without simulating individual scattering events.

To evaluate the the grain GSDFs (1), one needs to obtain α_{g}^{0} that depends solely on the shape of a grain as well as α_{g}^{+}, $p_{\mathrm{g}}^{\boldsymbol{x}}$, and $p_{\mathrm{g}}^{\boldsymbol{\omega}}$ that depend also on the grain's optical properties. To estimate these parameters, a Monte Carlo process analogous to volumetric path tracing was previously utilized in a per-grain (or per-grain-type) basis. Unfortunately, this becomes impractical when the grain optical properties are richly diverse or vary continuously.

To address this problem and allow precomputed GSDFs to be efficiently evaluated at render time with varying grain optical properties, we represent GSDFs as functions of not only incident and outgoing locations and directions but also grain albedo and optical density. Precisely, we express the GSDF values as *polynomials* of grain albedos and interpolate between varying densities using *first-order approximation*.

We now provide more details on how our GSDFs are precomputed (§3.2.1) and used at render time (§3.2.2).

3.2.1. GSDF Precomputation

We introduce a *symbolic* and *differentiated* Monte Carlo process (Algorithm 1) to precompute our GSDFs. Specifically, for each type of grain (with fixed shape and boundary BSDF), we express $\alpha_{\mathrm{g}}^{+}(\beta_{\mathrm{o}} \,|\, \sigma, a)$, $p_{\mathrm{g}}^{\boldsymbol{x}}(\boldsymbol{x}_{\mathrm{i}}, \beta_{\mathrm{o}} \,|\, \sigma, a)$, and $p_{\mathrm{g}}^{\boldsymbol{\omega}}(\boldsymbol{\omega}_{\mathrm{i}}, \beta_{\mathrm{o}} \,|\, \sigma, a)$ in Eq. (1) as functions of the grain's optical density σ and single-scattering albedo a, yielding:

$$f_{\mathrm{g}}(\boldsymbol{x}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{x}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{o}} \,|\, \sigma, a) \approx \alpha_{\mathrm{g}}^{0}(\beta_{\mathrm{o}})\, \delta(\boldsymbol{\omega}_{\mathrm{i}} + \boldsymbol{\omega}_{\mathrm{o}}) + \alpha_{\mathrm{g}}^{+}(\beta_{\mathrm{o}} \,|\, \sigma, a)\, p_{\mathrm{g}}^{\boldsymbol{x}}(\boldsymbol{x}_{\mathrm{i}}, \beta_{\mathrm{o}} \,|\, \sigma, a)\, p_{\mathrm{g}}^{\boldsymbol{\omega}}(\boldsymbol{\omega}_{\mathrm{i}}, \beta_{\mathrm{o}} \,|\, \sigma, a). \tag{10}$$

We now describe how α_{g}^{+}, $p_{\mathrm{g}}^{\boldsymbol{x}}$, and $p_{\mathrm{g}}^{\boldsymbol{\omega}}$ in Eq. (10) are represented and precomputed. Due to their similarities, we will focus on α_{g}^{+}. Recall that $\alpha_{\mathrm{g}}^{+}(\beta_{\mathrm{o}} \,|\, \sigma, a)$ captures the fraction of light with incident angle β_{o} to leave the grain after interacting with it. We describe α_{g}^{+} as a polynomial of grain albedo a of degree-K with coefficients $c_0^{\alpha}, \ldots, c_K^{\alpha}$:

$$\alpha_{\mathrm{g}}^{+}(\beta_{\mathrm{o}} \,|\, \sigma, a) = \sum_{k=0}^{K} c_k^{\alpha}(\beta_{\mathrm{o}}, \sigma)\, a^k = \underbrace{\begin{pmatrix} c_0^{\alpha}(\beta_{\mathrm{o}}, \sigma) \\ \vdots \\ c_K^{\alpha}(\beta_{\mathrm{o}}, \sigma) \end{pmatrix}}_{=:\, \boldsymbol{c}^{\alpha}(\beta_{\mathrm{o}}, \sigma)} \cdot \underbrace{\begin{pmatrix} 1 \\ \vdots \\ a^K \end{pmatrix}}_{=:\, \boldsymbol{a}}, \tag{11}$$

Algorithm 1 Estimating α_g^+ using symbolic and differentiated PT

1: **function** EstimateAlphaG+(β_o, σ)
2: $\boldsymbol{c}^\alpha \leftarrow \{0\}^{K+1}$, $\partial\boldsymbol{c}^\alpha \leftarrow \{0\}^{K+1}$
3: Randomly rotate the grain
4: Initialize a ray $(\boldsymbol{r}, \boldsymbol{\omega})$ with incident angle β_o
5: **if** the ray intersects the actual grain **then**
6: $T \leftarrow 1$ ▷ Path throughput
7: $k \leftarrow 0$ ▷ Number of scatterings
8: $\ell \leftarrow 0$ ▷ Distance traveled inside the grain
9: **while** ray $(\boldsymbol{r}, \boldsymbol{\omega})$ intersects the grain at $\boldsymbol{r}'$ **do**
10: **if** line segment $(\boldsymbol{r}, \boldsymbol{r}')$ lies inside the grain **then**
11: $t \leftarrow -\log(\text{rand}())/\sigma$
12: $\ell \leftarrow \ell + \min(t, \|\boldsymbol{r}' - \boldsymbol{r}\|)$
13: **else**
14: $t \leftarrow \infty$
15: **end if**
16: **if** $t < \|\boldsymbol{r}' - \boldsymbol{r}\|$ **then** ▷ Volumetric scattering
17: $k \leftarrow k + 1$
18: $\boldsymbol{r}' \leftarrow \boldsymbol{r} + t\boldsymbol{\omega}$
19: Draw a direction $\boldsymbol{\omega}'$ based on the grain's phase function
20: **else** ▷ Interfacial scattering
21: Draw a direction $\boldsymbol{\omega}'$ based on the grain's BSDF and scale T accordingly
22: **end if**
23: $\boldsymbol{r} \leftarrow \boldsymbol{r}'$, $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega}'$
24: **end while**
25: $\boldsymbol{c}^\alpha[k] \leftarrow T$
26: $\partial\boldsymbol{c}^\alpha[k] \leftarrow (k/\sigma - \ell)T$ ▷ Diff. throughput w.r.t. σ
27: **end if**
28: **return** $\boldsymbol{c}^\alpha$, $\partial\boldsymbol{c}^\alpha$
29: **end function**

where the coefficients c_k^α are functions of β_o and σ, and the "$\cdot$" symbol indicates vector inner product. Note that, when σ and a are both high, K needs to be large to avoid noticeable energy loss. We will discuss how this can be handled in §3.4.

Given β_o, with the grain optical density σ fixed, we estimate the coefficient vector $\boldsymbol{c}^\alpha \in \mathbb{R}^{K+1}$ in Eq. (11) using a modified path tracing technique that is *symbolic* with respect to grain albedo a. In other words, this process returns (the coefficients of) a polynomial of a. A similar process was used previously by Hašan and Ramamoorthi [HR13].

Unlike conventional path tracing where the path throughput is scaled by the single-scattering albedo a whenever light scatters within the grain, our symbolic path tracer does not scale the throughput at these scattering events. Instead, it records the total number of scatterings on each light transport path. When the light eventually leaves the grain after k scatterings, the path tracer returns a vector with its k-th component setting to the final path throughput T and all the others to zero. Lastly, $\boldsymbol{c}^\alpha(\beta_o, \sigma)$ equals the expected value of the symbolic path tracing output. Please see Algorithm 1 for a summary of this process.

In Eq. (11), the grain optical density σ is assumed to be fixed. To allow this parameter to vary continuously at render time without excessive precomputation, we leverage a first-order approximation:

$$\alpha_g^+(\beta_o \,|\, \sigma, a) \approx \alpha_g^+(\beta_o \,|\, \sigma_0, a) + (\sigma - \sigma_0)\,\partial\alpha_g^+(\beta_o \,|\, \sigma_0, a), \quad (12)$$

where

$$\partial\alpha_g^+(\beta_o \,|\, \sigma_0, a) := \frac{\partial\alpha_g^+}{\partial\sigma}(\beta_o \,|\, \sigma_0, a) = \underbrace{\begin{pmatrix} \frac{\partial c_0^\alpha}{\partial\sigma}(\beta_o, \sigma_0) \\ \vdots \\ \frac{\partial c_{K-1}^\alpha}{\partial\sigma}(\beta_o, \sigma_0) \end{pmatrix}}_{=:\,\partial\boldsymbol{c}^\alpha(\beta_o, \sigma_0)} \cdot \boldsymbol{a}, \quad (13)$$

is the partial derivative of Eq. (11) with respect to σ evaluated at some fixed expansion point σ_0.

To estimate $\partial\boldsymbol{c}^\alpha$ in Eq. (13), we slightly extend our symbolic path tracing algorithm (Line 26 of Algorithm 1). Please refer to §1 in the supplemental document for detailed derivations. Lastly, combining Eqs. (11–13) yields

$$\alpha_g^+(\beta_o \,|\, \sigma, a) \approx \left[\boldsymbol{c}^\alpha(\beta_o, \sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{c}^\alpha(\beta_o, \sigma_0)\right] \cdot \boldsymbol{a}. \quad (14)$$

The other two terms $p_g^{\boldsymbol{x}}$ and $p_g^{\boldsymbol{\omega}}$ in Eq. (10), which are generally represented as tabulated (i.e., piecewise constant) functions of $\boldsymbol{x}_i$ and $\boldsymbol{\omega}_i$, can be estimated similarly via

$$\boldsymbol{p}_g^{\boldsymbol{x}}(\beta_o \,|\, \sigma, a) \approx \left[\boldsymbol{C}^{\boldsymbol{x}}(\beta_o, \sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{C}^{\boldsymbol{x}}(\beta_o, \sigma_0)\right] \boldsymbol{a}, \quad (15)$$

$$\boldsymbol{p}_g^{\boldsymbol{\omega}}(\beta_o \,|\, \sigma, a) \approx \left[\boldsymbol{C}^{\boldsymbol{\omega}}(\beta_o, \sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{C}^{\boldsymbol{\omega}}(\beta_o, \sigma_0)\right] \boldsymbol{a}, \quad (16)$$

where $\boldsymbol{C}^{\boldsymbol{x}}$, $\partial\boldsymbol{C}^{\boldsymbol{x}}$, $\boldsymbol{C}^{\boldsymbol{\omega}}$, $\partial\boldsymbol{C}^{\boldsymbol{\omega}} \in \mathbb{R}^{m\times(K+1)}$ are coefficient matrices and their derivatives that can be estimated using similar processes analogous to Algorithm 1.

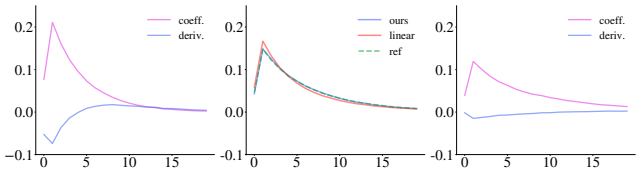
In practice, for each β_o, we precompute $\boldsymbol{c}^\alpha$, $\partial\boldsymbol{c}^\alpha$, $\boldsymbol{C}^{\boldsymbol{x}}$, $\partial\boldsymbol{C}^{\boldsymbol{x}}$, $\boldsymbol{C}^{\boldsymbol{\omega}}$ and $\partial\boldsymbol{C}^{\boldsymbol{\omega}}$ at a set of predetermined expansion points $\{\sigma_1, \sigma_2, \ldots\}$. Further, we pick $m = 180^2$ for $p_g^{\boldsymbol{x}}$ and 360×180 for $p_g^{\boldsymbol{\omega}}$. The choice of K will be discussed in §3.4. At render time, given any grain optical density σ, we apply Dutch Taylor expansion [Kra03] using the two nearest expansion densities. This is achieve by evaluating Eqs. (14–16) on them, and linearly interpolating the results (see Figure 3 for an example).

Other optical parameters. Our symbolic and differentiable tracing method described in Algorithm 1 can be easily extended to support other optical properties such as boundary BSDF and single-scattering phase function as long as they are parameterized in a way that can be symbolically differentiated.

3.2.2. GSDF Sampling at Render Time

To efficiently use our precomputed GSDFs specified by Eqs. (10, 14–16) at render time, we need to importance sample them.

Consider the joint sampling of $\boldsymbol{x}_i$ and $\boldsymbol{\omega}_i$ given $\boldsymbol{x}_o$ and $\boldsymbol{\omega}_o$. Provided arbitrary grain density σ and single-scattering albedo a, we first calculate $\alpha_g^+(\beta_o \,|\, \sigma, a)$ via Eq. (14) in $O(K)$ time. Then, we draw a random number $\xi \sim U[0,1)$ that in turn leads to one of three possible outcomes: (i) the ray $(\boldsymbol{x}_o, -\boldsymbol{\omega}_o)$ misses the actual grain and keeps going forward (when $\xi < \alpha_g^0$); (ii) the ray interacts with the grain (when $\alpha_g^0 \le \xi < \alpha_g^0 + \alpha_g^+(\beta_o \,|\, \sigma, a)$) without being absorbed; and (iii) the ray gets absorbed by the grain (when $\xi \ge \alpha_g^0 + \alpha_g^+(\beta_o \,|\, \sigma, a)$).



(a) $\sigma = 2.15$ (b) $\sigma = 3.35$ (c) $\sigma = 4.64$

Figure 3: Render-time GSDF evaluation. Given the grain optical density σ and albedo a, we evaluate the GSDF functions α_g^+, $\boldsymbol{p}_g^{\boldsymbol{x}}$, and $\boldsymbol{p}_g^{\boldsymbol{\omega}}$ at the render time by computing first-order approximations (14–16) using two precomputed expansion values closest to σ and linearly interpolate the results. This examples shows the evaluation of α_g^+ for one fixed β_o where each curve illustrate a coefficient vector $\boldsymbol{c}^\alpha$ or $\partial\boldsymbol{c}^\alpha$. In (a) and (c), we show precomputed $\boldsymbol{c}^\alpha$ or $\partial\boldsymbol{c}^\alpha$ at two expansion locations of σ. In (b), at $\sigma = 3.35$, we compare our first-order approximation of $\boldsymbol{c}^\alpha$ (in blue), the reference (in green), and a simple linear interpolation obtained using only the precomputed $\boldsymbol{c}^\alpha$ from (a) and (c). Our result matches the reference perfectly.

In the second case, to obtain $\boldsymbol{x}_i$ and $\boldsymbol{\omega}_i$, we need to further sample the GSDF functions $p_g^{\boldsymbol{x}}$ and $p_g^{\boldsymbol{\omega}}$, respectively. This generally boils down to sampling discrete distributions determined by their tabulated counterparts $\boldsymbol{p}_g^{\boldsymbol{x}}$, $\boldsymbol{p}_g^{\boldsymbol{\omega}} \in \mathbb{R}^m$ given by Eqs. (15, 16). Unfortunately, fully computing these vectors for every sampling operation is costly as it involves expensive matrix-vector multiplications. We instead leverage a new strategy to efficiently sample $p_g^{\boldsymbol{x}}$ and $p_g^{\boldsymbol{\omega}}$.

We now focus on sampling $p_g^{\boldsymbol{x}}$, and $p_g^{\boldsymbol{\omega}}$ can be handled in a similar fashion. Let $\boldsymbol{P}_g^{\boldsymbol{x}}$ be an $m \times (K+1)$-matrix given by

$$\boldsymbol{P}_g^{\boldsymbol{x}}(\beta_o \,|\, \sigma, a) := \boldsymbol{Q}_g^{\boldsymbol{x}}(\beta_o \,|\, \sigma) \begin{pmatrix} 1 & & & \\ & a & & \\ & & \ddots & \\ & & & a^K \end{pmatrix},$$
$$\text{where} \quad \boldsymbol{Q}_g^{\boldsymbol{x}}(\beta_o \,|\, \sigma) := \boldsymbol{C}^{\boldsymbol{x}}(\beta_o, \sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{C}^{\boldsymbol{x}}(\beta_o, \sigma_0). \tag{17}$$

According to Eqs . (15, 17), it is easy to verify that the j-th component of $\boldsymbol{p}_g^{\boldsymbol{x}} \in \mathbb{R}^m$ equals the sum of all elements in the j-th row of $\boldsymbol{P}_g^{\boldsymbol{x}} \in \mathbb{R}^{m\times(K+1)}$. Thus, sampling j with a probability mass proportional to $\boldsymbol{p}_g^{\boldsymbol{x}}$ can be achieved by drawing an element $(j,k) \in \{0,\ldots,m-1\} \times \{0,\ldots,K\}$ from $\boldsymbol{P}_g^{\boldsymbol{x}}$ (with a probability proportional to the element values) and returning the row index j.

We leverage a matrix row-column sampling schemeto importance sample $\boldsymbol{P}_g^{\boldsymbol{x}}$ efficiently. Specifically, we first draw a column k with a probability proportional to $\mathrm{sum}(\boldsymbol{P}_g^{\boldsymbol{x}}[:,k])$, the sum of all elements in the k-th column of $\boldsymbol{P}_g^{\boldsymbol{x}}$ which in turn equals

$$\mathrm{sum}(\boldsymbol{P}_g^{\boldsymbol{x}}[:,k]) = \left[\mathrm{sum}(\boldsymbol{C}^{\boldsymbol{x}}[:,k]) + (\sigma - \sigma_0)\,\mathrm{sum}(\partial\boldsymbol{C}^{\boldsymbol{x}}[:,k])\right] a^k. \tag{18}$$

To compute Eq. (18) efficiently, we precompute and store the column sums of coefficient matrices $\boldsymbol{C}^{\boldsymbol{x}}$ and $\partial\boldsymbol{C}^{\boldsymbol{x}}$. This allows the sampling of k to be accomplished in $O(K)$ time.

After k is drawn, we sample a row j with a probability proportional to $\boldsymbol{Q}_g^{\boldsymbol{x}}[j,k]$ in $O(m)$ time. Lastly, $\boldsymbol{x}_i$ is drawn uniformly from $\mathbb{S}_j$, the j-th subdomain of $\mathbb{S}^2$ (provided by the tabulation of $p_g^{\boldsymbol{x}}$). The full sampling process runs in $O(K+m)$ time and is summarized in Algorithm 2 in the supplemental document.

3.3. Medium-Based Modeling

As stated in §3.1.2, granular materials can be modeled as continuous participating media at physical scales where individual grains cannot be resolved. We follow the formulation introduced in §3.2 by expressing the medium-level parameters σ^M and $\hat{f}^M$ as polynomials of grain albedo a evaluated at expansion densities $\{\sigma\}$.

Precomputing and sampling medium parameters. Obtaining the medium scattering properties σ^M, a^M, and f^M of a voxel via Eqs. (3–8) requires the grain-level mean ray offsets $\hat{\lambda}^\delta$ and $\hat{\lambda}^s$ as well as the scaled phase function $\hat{f}$ and effective albedo $\hat{\alpha}^s$. Except $\hat{\lambda}^\delta$ which only relies on a grain's shape, all the other parameters depend on the grain's optical properties σ and a.

To allow efficient precomputation and rendering, we leverage the same formulation used for GSDFs (§3.2.1) by representing the mean ray offset $\hat{\lambda}^s$ and scaled phase function $\hat{f}$ as

$$\hat{\lambda}^s(\sigma, a) \approx \left[\boldsymbol{c}^{\hat{\lambda}^s}(\sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{c}^{\hat{\lambda}^s}(\sigma_0)\right] \cdot \boldsymbol{a}, \tag{19}$$

$$\hat{f}(x \,|\, \sigma, a) \approx \left[\boldsymbol{C}^{\hat{f}}(x, \sigma_0) + (\sigma - \sigma_0)\,\partial\boldsymbol{C}^{\hat{f}}(x, \sigma_0)\right] \boldsymbol{a}. \tag{20}$$

In Eqs. (19, 20), the coefficient vectors $\boldsymbol{c}^{\hat{\lambda}^s}, \partial\boldsymbol{c}^{\hat{\lambda}^s} \in \mathbb{R}^K$ and matrices $\boldsymbol{C}^{\hat{f}}, \partial\boldsymbol{C}^{\hat{f}} \in \mathbb{R}^{m\times K}$ can be estimated using an adapted version of our symbolic and differentiated path tracing (Algorithm 1).

In practice, for each type of grain (with fixed shape and surface BSDF), we precompute $\hat{\lambda}^\delta$ (that is independent of grain scattering properties) as well as $\boldsymbol{c}^{\hat{\lambda}^s}$, $\partial\boldsymbol{c}^{\hat{\lambda}^s}$, $\boldsymbol{C}^{\hat{f}}$, and $\partial\boldsymbol{C}^{\hat{f}}$ at a set of predetermined σ values $\boldsymbol{\Sigma} := \{\sigma_1, \sigma_2, \ldots\}$.

At render time, when using VPT, we sample free-flight distances using delta tracking [WMHL65] which only requires per-point evaluations of σ^M using Eqs. (3–7, 19). When sampling medium scattering and absorption using the scaled phase function $\hat{f}^M$, we first pick a grain n uniformly at random and then sample its effective scaled phase function $\hat{f}_n$ using a matrix row-column sampling scheme similar to §3.2.2.

Multi-scale rendering. We have described three methods to render a granular material. Explicit path tracing (EPT) can be used to render the material with each grain fully described. This method offers the best fidelity but is also the most expensive. Proxy path tracing (PPT) uses GSDFs to speed up the rendering of individual grains at the cost of losing grain-level details. This approach is particularly useful in handling highly scattering grains. Lastly, volume path tracing (VPT) renders the material as a continuous medium and is best suited for large-scale problems where individual grains are invisible.

We switch between these methods on the fly using heuristics similar to those introduced by prior works [MPH*15, MPG*16]. Specifically, we switch from EPT to PPT after light interacting with the grain at least once. When deciding whether to switch to VPT,

we consider distances between current scattering locations (of a set of simultaneously traced light transport paths) to the surface of the medium as well as the variance of these locations. Notice that these heuristics are entirely orthogonal to our technique and can be replaced with more sophisticated ones if needed.

Beyond VPT. When storing the precomputed medium scattering (i.e., radiative transfer) parameters in a voxelized fashion, the granular material, as a standard heterogeneous participating medium, can then be rendered using existing techniques such as precomputed shell transport [MWM07, MPG*16] and diffusion methods [AWB11, MPH*15]) that are more efficient than volume path tracing (VPT). Notice that storing full medium-level models as 3D volumes is usually practical (as opposed to realizing all grains) because individual voxels generally contain large quantities of grains. As the use of shell transport/diffusion methods is orthogonal to our technique, we use VPT to render our models at the coarsest level.

3.4. Supporting Highly Scattering Grains

In §3.2 and §3.3, for fixed grain density σ, we used degree-K polynomials of the grain albedo a to symbolically express the GSDF functions α_g^+, $p_g^{\boldsymbol{x}}$, $p_g^{\boldsymbol{\omega}}$, the radiative transfer parameters $\hat{\lambda}^s$, $\hat{f}$, and their derivatives. To support highly-scattering grains (i.e., those with high optical density σ and single-scattering albedo a), high-order polynomials are needed to avoid noticeable energy loss. Unfortunately, this yields not only larger data footprints but also slower rendering due to more expensive polynomial evaluations.

To address this problem, we extend our polynomials in a way that, instead of simply neglecting all coefficients beyond some fixed degree K, we approximate them analytically so that the resulting polynomials effectively have infinite degrees.

Extended polynomials. According to our experiments, the polynomial coefficients $\{c_k\}$ for all the GSDF and radiative transfer parameters decrease in a roughly exponential fashion when k is large. In other words, there exists some $\tau > 0$ such that, for all $k > K$,

$$c_k \approx c_K \, \mathrm{e}^{\tau(K-k)}. \tag{21}$$

Figure 4-a illustrates such an example. Then, by storing $c_0, c_1, \ldots, c_K$ and τ, we effectively obtain a polynomial that has infinite degree and can be evaluated efficiently via

$$\sum_{k=K+1}^{\infty} c_k a^k = c_K \sum_{n=1}^{\infty} \left(\mathrm{e}^{-\tau} a\right)^n = \frac{\mathrm{e}^{-\tau} a}{1 - \mathrm{e}^{-\tau} a}. \tag{22}$$

To acquire τ, we precompute the polynomial coefficients $\{c_k\}$ up to some degree $K' \gg K$ and solve a least square problem in the logarithmic space. This gives

$$\tau = \frac{\sum_{n=1}^{K'-K} (n \Delta c_n)^2}{\sum_{n=1}^{K'-K} n^2}, \quad \text{where } \Delta c_n := \log(c_K) - \log(c_{K+n}). \tag{23}$$

Please see §2 in the supplemental document for the derivation.

Recall that, to obtain the coefficient $c_k(\sigma)$ for arbitrary grain density σ at render time, we used first-order approximations (14–16, 19, 20) for $k \le K$. Although this can be done for the extended coefficients (i.e., c_k with $k > K$) as well, which is discussed in §2 in the supplemental document, we found it unnecessary. This is because, when k is large, ∂c_k is generally so small that simple linear interpolations of c_k works adequately. That is, we can express $c_k(\sigma)$ by linearly interpolating $c_k(\sigma_0)$ and $c_k(\sigma_1)$, the values of c_k precomputed at the neighboring expansion points σ_0 and σ_1 (with $\sigma_0 \le \sigma \le \sigma_1$). Precisely, for all $k > K$,

$$\begin{aligned} c_k(\sigma) &\approx \tfrac{\sigma_1 - \sigma}{\sigma_1 - \sigma_0} c_k(\sigma_0) + \tfrac{\sigma - \sigma_0}{\sigma_1 - \sigma_0} c_k(\sigma_1) \\ &= \tfrac{\sigma_1 - \sigma}{\sigma_1 - \sigma_0} c_K(\sigma_0) \, \mathrm{e}^{\tau(\sigma_0)(K-k)} + \tfrac{\sigma - \sigma_0}{\sigma_1 - \sigma_0} c_K(\sigma_1) \, \mathrm{e}^{\tau(\sigma_1)(K-k)}. \end{aligned} \tag{24}$$

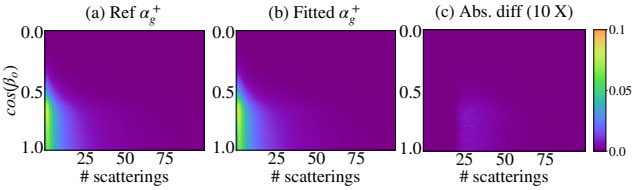


Figure 4: The exponential falloff of polynomial coefficients. In (a), we illustrate a GSDF function $\alpha_g^+(\beta_o)$ where each row of the image illustrates a coefficient vector $\boldsymbol{c}^\alpha$ with a fixed β_o. At $\sigma = 10$, extended polynomials with degree 20 can accurately capture the long tails of all the coefficient vectors, as shown in (b) and (c).

Energy conservation. To ensure that our GSDFs and medium scattering parameters represented using extended polynomials conserve energy, we simply clamp the corresponding albedo values $(\alpha_g^0 + \alpha_g^+)$ and $\hat{\alpha}^s$ to one. In practice, this clamping occurs very rarely thanks to the accurate approximation provided by Eq. (21).

Sampling extended polynomials. In §3.2, we introduced a matrix row-column sampling scheme to efficiently sample the polynomial-valued GSDF $p_g^{\boldsymbol{x}}$, $p_g^{\boldsymbol{\omega}}$ and scaled phase function $\hat{f}^M$. That is, to sample a discrete distribution with m outcomes each with a probability given by a degree-K polynomial, we construct an $m \times (K+1)$-matrix $\boldsymbol{P}$ using the polynomial weights. Then, we first draw a column of $\boldsymbol{P}$, and then draw a row from that column.

To support our extended polynomials, we add a new column to the matrix $\boldsymbol{P}$ that captures the polynomial coefficients described analytically via Eq. (21). Given the grain optical properties σ and a, elements in this additional column can be calculated using Eqs. (22, 24). During the sampling process at render time, we can first compute the elements in the added column, obtain their sum, and apply the row-column sampling scheme to the new matrix with $(m+1)$ columns. In this way, the entire sampling process still takes $O(m+K)$ time.

To further improve the sampling performance, we approximate the sum of all elements from the added column using another exponential function, removing the need to iterate over each element of the additional column in each sampling step. This is particularly helpful when the grain albedo is low since it avoids wasting time evaluating the element values given that the added column is unlikely to be sampled in the first place. Although the extra approximation could reduce result accuracy, we do not observe any visible difference when using approximated column sums in practice.

4. Tile-Based Modeling of Granular Materials

Previously, individual grains in a granular material are either explicitly stored [MPG*16] or obtained by replicating a single tile [MPH*15]. These methods have difficulties in handling large-scale materials with non-uniform grain properties and distributions. To address this problem, we introduce a new tile-based representation for granular materials. This model describes the grains using a three-dimensional grid of tiles that can be realized at render time (§4.1). Then, in §4.2, we show how this model can be rendered using our multi-scale framework described in §3.

4.1. Tile-Based Representation

Under our tile-based representation, a granular material is depicted via a set of *tiles*. Each tile $\mathcal{T}_j$, in turn, is a three-dimensional cell that records the locations of a few uniformly distributed grains. The number of grains can vary between tiles, allowing the modeling of large-scale nonuniform grain distributions. Notice that our tiles only specifies the grain locations. Other attributes including the grains' shapes, sizes, and optical properties are specified separately via a process detailed later in this subsection.

Provided the desired number of grains $\mathcal{N}_j$ in each tile $\mathcal{T}_j$, we aim to represent and construct all the tiles efficiently without the need to store the locations for individual grains.

Template tile. At the core of our tile-based model is a *template tile* $\mathcal{T}$ that stores all possible in-tile grain locations $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots \in \mathbb{R}^3$ which are then reused to instantiate all the tiles. Specifically, each tile $\mathcal{T}_j$ is obtained by selecting a subset of $\mathcal{N}_j$ grain locations from the template.

To this end, we sort the grain centers in the template tile so that any prefix $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k\}$ are distributed evenly. Additionally, for each prefix with k centers, we calculate and store the maximal radius $\tilde{R}_k$ for k spheres located at the center locations to have no intersection. Namely,

$$\tilde{R}_k := \sup\{r : B(\boldsymbol{y}_i, r) \cap B(\boldsymbol{y}_j, r) = \emptyset \quad \text{for all } 1 \le i < j \le k\}, \tag{25}$$

where $B(\boldsymbol{y}, r) \subset \mathbb{R}^3$ denotes the sphere centered at $\boldsymbol{y}$ with radius r.

Tile precomputation. We use a technique developed in computational physics [SDST06] to generate the grain centers in our template tile. This technique offers a highly packing rate while preserving visual realism (by avoiding visible repetitive patterns).

To rank the generated centers in the template tile, we leverage a dart-throwing based technique that was originally developed in Poisson-disc sampling [KCODL06, Yuk15] and works as follows. Given a set of pre-generated center locations, a random one is chosen as $\boldsymbol{y}_1$. Then, for $i = 1, 2, \ldots$, the next grain center $\boldsymbol{y}_{i+1}$ is picked (among the unused centers) such that the distance to all chosen centers (tiled infinitely) is maximized.

Notice that, for Poisson-disc sampling, multiple template tiles are normally used to avoid repetitive sample patterns. Although our model representation is fully compatible with multiple such tiles, we use only one in practice since the variations in grain shape, orientation, and appearance, when combined with our added randomnesses (see the discussions below), make periodic center locations virtually impossible to perceive.

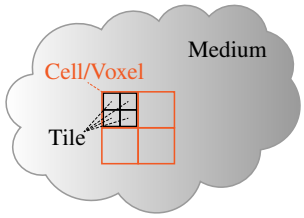


Figure 5: An illustration of our **tile-based** representation which expresses grains with complex nonuniform spatial distribution using a 3D grid in which each cell, or voxel, is in turn comprised of a number of tiles.

Introducing randomness. For tiles with identical grain counts, always arranging the grains at the same center locations from the template tile can result in visible repetitive patterns. To address this problem, we introduce randomnesses to the model.

For a tile $\mathcal{T}_j$ that needs $\mathcal{N}_j$ grains with some desired radius r_j, let $n = \max\{k : \tilde{R}_k \ge r_j\}$. Then, spheres centered at $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ with radii r_j are guaranteed to be intersection-free. Thus, we can pick any subset containing $\mathcal{N}_j$ elements as the grain center locations for tile $\mathcal{T}_j$, allowing additional randomness to be introduced.

In practice, we randomly choose a contiguous subsequence of $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ with $\mathcal{N}_j$ elements. To be precise, we draw an integer t, the index of the first element in the subsequence, from $\{1, 2, \ldots, n - \mathcal{N}_j + 1\}$ uniformly at random. The grain centers $\boldsymbol{y}_t$, $\boldsymbol{y}_{t+1}$, …, $\boldsymbol{y}_{t+\mathcal{N}_j-1}$ are then assigned to tile $\mathcal{T}_j$. Here we use contiguous subsequences (instead general ones) since they can be randomly selected very efficiently while providing sufficient visual variation.

Beyond tiles. To specify the number of grains $\mathcal{N}_j$ for each tile $\mathcal{T}_j$, we use a regular 3D grid in which each cell contains multiple tiles (see Figure 5). By specifying the desired grain count at each grid vertex, we can then obtain the grain density $\rho(\boldsymbol{x})$ for any location $\boldsymbol{x}$ via tri interpolation. Then, we have

$$\mathcal{N}_j = \left\lfloor \int_{\mathcal{T}_j} \rho(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \right\rfloor. \tag{26}$$

The grain radius r_j can be specified similarly using another 3D grid.

Specifying grain properties. With the grain locations and radii expressed, we now describe how the properties of each grain are specified. We assume the granular material to contain a few types of grains such that those of the same type share identical shapes and surface BSDFs but can have varying scattering parameters (i.e., optical density σ and single-scattering albedo a).

Based on this assumption, we use a 3D grid that stores at each vertex a weight w_i for every grain type i. Then, at render time, we randomly assign a type i to each grain in tile $\mathcal{T}_j$ with a probability:

$$p_i := \frac{\int_{\mathcal{T}_j} w_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}{\sum_i \int_{\mathcal{T}_j} w_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}, \tag{27}$$

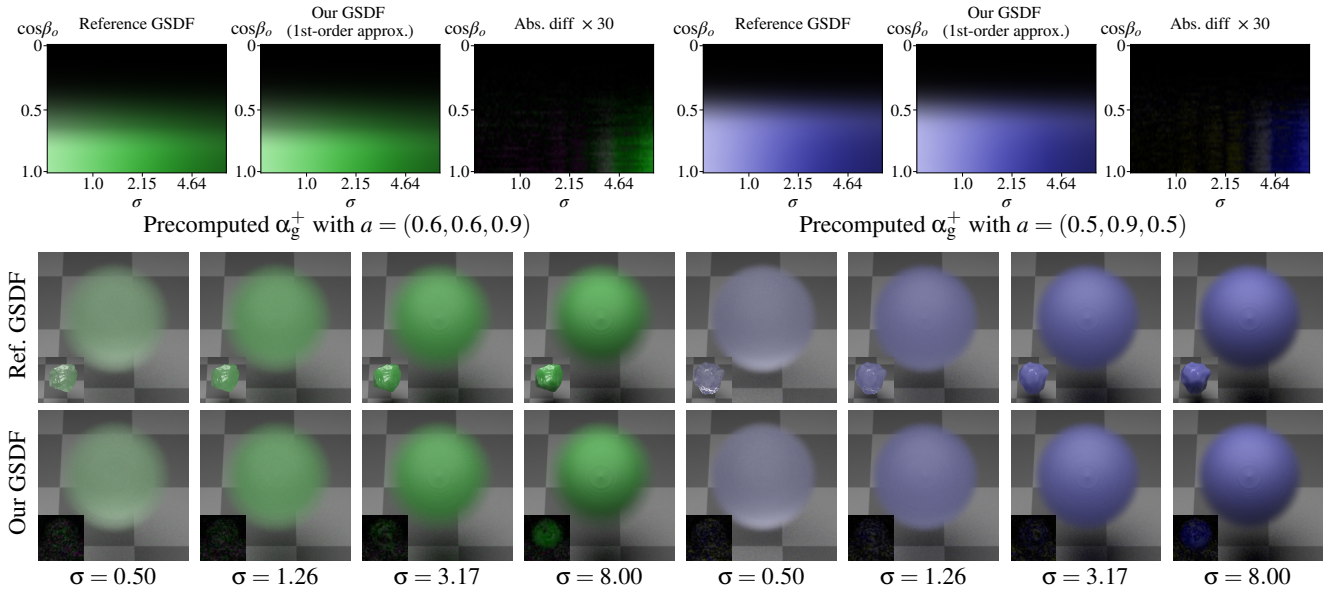


Figure 6: Our **precomputed GSDFs** obtained via Algorithm 1 allow grain optical properties σ and a to be varied smoothly at render time. In this example, we illustrate the α_g^+ components of two GSDFs evaluated at fixed albedo values (top). The ticks at the $x-$axis indicate the expansion locations where the derivatives are precomputed and stored. Additionally, we show grain renderings generated using reference and our GSDFs with identical albedo a and varying optical densities σ (bottom). Renderings of grains with fixed orientations and $8\times$-absolute error images (containing mostly Monte Carlo noise) are shown as insets.

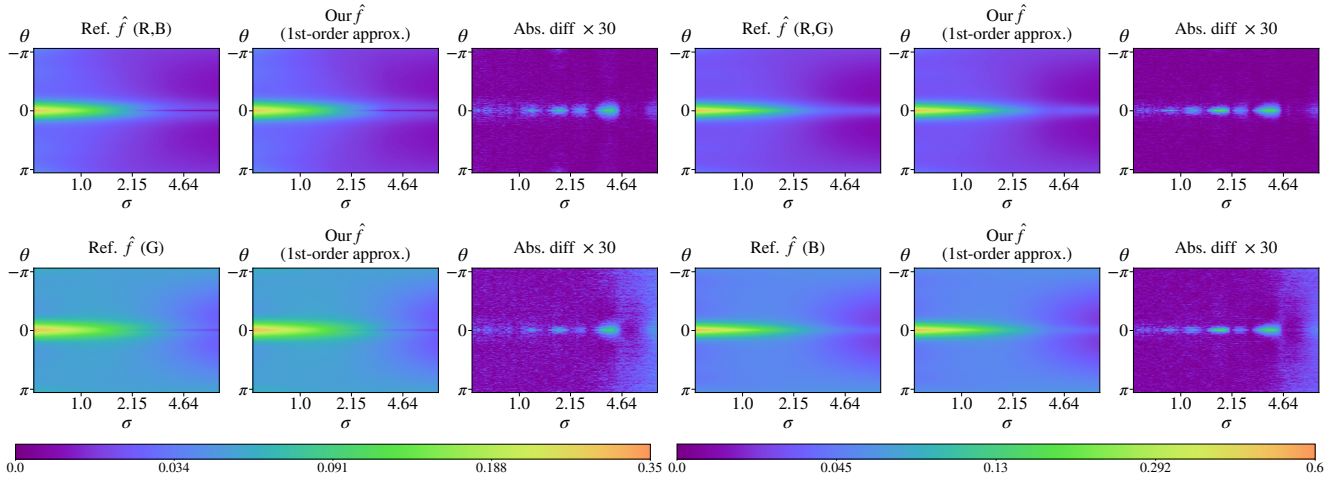


Figure 7: Scaled phase functions $\hat{f}$ at the grain level determine a granular material's medium scattering parameters σ^{M} and $\hat{f}^{\mathrm{M}}$ via Eqs. (3–9). In this example, we validate our scaled phase functions (which allows grain optical properties to vary at render time) precomputed for the two grains used in Figure 6.

where $w_i(\boldsymbol{x})$ is obtained by linearly interpolating the corresponding weights at the grid vertices. Lastly, the optical properties of each grain are specified by evaluating two user-provided functions $\sigma(\boldsymbol{x})$ and $a(\boldsymbol{x})$, which can be expressed in piecewise fashions or procedurally, at the center of the grain.

Medium scattering parameters. Our tile-based models can be approximated as continuous participating media using the approach described in §3.3. In this case, we treat each tile $\mathcal{T}_j$ as a voxel and compute its optical density σ^{M} and scaled phase function $\hat{f}^{\mathrm{M}}$ on the fly. Notice that the summations over all grains in the voxel (6–8) can be computed efficiently as weighted sums over each grain type.

4.2. Ray-Tracing Tile-Based Granular Materials

The rendering techniques introduced in §3 are compatible with our tile-based representation of granular materials.

To render our tile-based model at the finest scale using explicit path tracing (EPT), the key operation is to compute the intersection between a given ray and all grain surfaces. Since the grains are not stored individually, we introduce a new algorithm to efficiently ray-trace our model. This algorithm works at two separate scales: cell and tile.

Computing ray-cell intersections. We precompute a bounding volume hierarchy (BVH) based on the 3D grids that specify the grain distributions $\mathcal{N}_i$ for each grain type i. Recall that each cell in this grid generally contains multiple tiles. The BVH's root node contains the entire grid while its leaf corresponds to individual cells. When splitting a node, we always follow the cell boundaries so that each cell belongs to exactly one node in any depth of the hierarchy. At render time, this BVH helps to quickly compute ray-cell intersections.

When the ray hits a cell, we need to check if it indeed intersects a grain inside. To this end, we subdivide the cell on the fly until reaching a single tile. Lastly, we use a customized Kd-tree precomputed for the template tile to efficiently identify if the ray hits any grain in this tile (see below for more discussions). The full render-time ray intersection computation is summarized in Algorithm 3 in the supplemental document.

Computing ray-tile intersections. When reaching a leaf of the precomputed BVH, we need to test if a given ray intersects any grains within the cell. As a grid cell is generally comprised of multiple tiles, this boils down to ray-trace individual tiles. To do this efficiently, we pre-construct a Kd-tree using the surface-area heuristics (SAH) for the template tile. This Kd-tree contains all grain centers as points in 3D, and its internal nodes also record the range of grain indices from the corresponding subtrees. When computing the intersection between the ray and an actual tile $\mathcal{T}_j$ specified with (i) an interval $[u,v]$ indicating the range of template grains are in this tile and (ii) the grain radii r_j, we traverse the precomputed Kd-tree from the root. During the traversal, we quickly prune all subtrees with constituent grains having indicies disjoint of $[u,v]$.

5. Results

We now demonstrate the effectiveness of our technique via a few examples. In §5.1, we validate our method by comparing GSDF renderings of individual grains. In §5.2, we show rendered results of a few heterogeneous granular materials with a range of grain distributions and properties at greatly varying physical scales.

We implemented our technique based on the Mitsuba physically based renderer [Jak18]. All our precomputed models are stored in uncompressed binary format using 16-bit floats.

5.1. Validations and Evaluations

In §3, we introduced a new symbolic and differentiated Monte Carlo process to precompute GSDFs and medium scattering parameters while allowing the grain optical properties to be continuously varied at render time. Figures 6 and 7 show validations of our precomputed GSDFs and grain-level scaled phase functions (which in turn determine the medium scattering parameters) for two example grains. Results obtained using our first-order approximations, i.e., Eqs. (14–16, 19, 20), closely match the references obtained with existing techniques [MPH*15, MPG*16] (that apply conventional Monte Carlo simulations to fixed combinations of grain optical properties). Notice that both the reference and our GSDF results approximate expected grain appearances (with random orientations).

Additionally, we demonstrate the effectiveness of our polynomial representation Eqs. (11) and (12) in Figure 8. This example involves a material of grains with gradually changing colors (Figure 8-a). A naïve solution is to discretize the grain optical properties (i.e., albedo and density) into a number of bins and precompute Disney GSDFs and medium properties in a per-bin basis. This, with similar total model sizes, can lead to noticeable discontinuities (as shown in Figure 8-b). Another baseline is to compute GSDFs by linearly interpolating those precomputed at a discrete set of grain properties (i.e., density and albedo values). This, however, yields limited physical accuracy (Figure 8-c) due to the non-linearity of GSDFs. The accuracy can be further increased if we depict symbolically in grain albedo using the polynomial representation Eq.(11) (Figure 8-d). Our method offers significantly better physical accuracy (see Figure 8-e).

Lastly, we provide a white furnace test to demonstrate the effectiveness of our extended polynomials formulation Eq. (21) in the supplemental document.

5.2. Main Results

We now show a few examples rendered with our technique. Detailed model size and render time statistics for all these examples are available in Table 1.

Models with explicitly described grains. Figures 9 and 10 contain renderings of granular materials with the grains described and stored individually. In Figures 9, we show a pile of 172 K grains with gradually changing colors, an effect that cannot be efficiently handled by prior methods. Thanks to our symbolic and differentiable path tracing process, our method is capable of efficiently rendering this model at high accuracy with minimal precomputation.

Figure 10 shows another example where 144 K translucent grains with smoothly varying optical densities fall through a funnel. An early and a late state of the grains during this process are shown in this figure, and our method successfully captures the smooth change of grain translucency nicely. Please refer to the accompanying video for the full animation.

Models using our tile-based representation. Figures 10 and 12 show examples modeled with our tile-based approach described in §4 and rendered with explicit path tracing (EPT) and the multi-scale approach (EPT+PPT+VPT) discussed in §3.

In Figure 11, we show a castle comprised of 4.5 billion highly scattering grains with similar optical properties as those in Figure

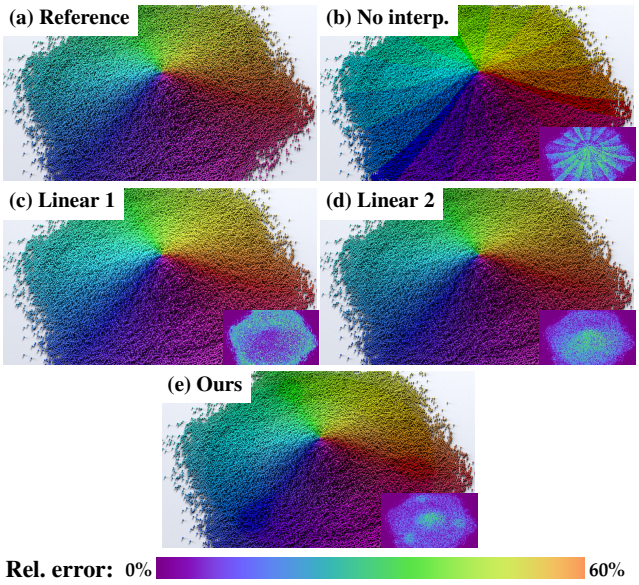


Figure 8: Effectiveness of our approach: when representing materials with gradually changing grain properties, using GSDFs and medium properties precomputed at coarsely discretized grain albedo and densities can lead to visually distracting artifacts (b). Depicting those properties by linearly interpolating between grain properties can provide visually plausible results but is limited in physical accuracy (c). The accuracy can be further increased if using the polynomial representation to depict the grain albedo (d). Our first-order-approximation-based method offers significantly better accuracy (e).

Table 1: Data size and performance statistics. Render times are measured in *time to unit variance* (TTUV) in a similar fashion as Meng et al. [MPH*15].

Scene		Grain count	Data size (GB) GSDF	Data size (GB) others	TTUV (sec.) ref.	TTUV (sec.) ours
Pile	(Fig. 9)	172 K	1.08	0.16	18.47	4.21
Funnel	(Fig. 10)	128 K	1.52	0.41	11.52	3.36
Castle	(Fig. 11)	4.5 B	6.51	0.33	838.96	131.42
Explosion	(Fig. 12)	10 B	2.6	0.58	82.56	1.6
Saturn	(Fig. 1, 12)	100 B	0.72	2.67	7.74	0.024

2 in the supplement. Thanks to our extended polynomial representation, our method manages to closely resemble the reference appearance while using simple truncated polynomial suffers from high energy loss.

Lastly, Figure 12 shows another two large-scale models expressed using our tile-based method. On the top of this figure, we show an explosion of 10 billion grains with spatially varying shapes, sizes, optical densities, and single-scattering albedo. Our technique enjoys the efficiency to model this challenging scene without storing individual grains. Further, multi-scale rendering enabled by our precomputation yields significantly better performance.

On the bottom of Figure 12, we show renderings of the Saturn

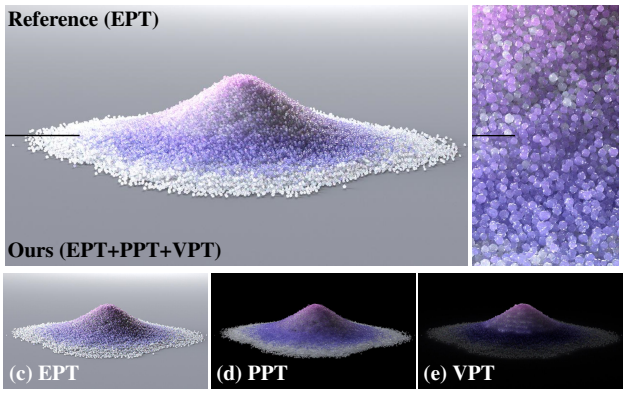


Figure 9: Component-wise evaluation of our technique. We enable multi-scale rendering of granular materials by switching between explicit path tracing (EPT), proxy path tracing (PPT), and volumetric path tracing (VPT) on the fly. This example shows the contribution of each component when rendering a pile of colorful grains.

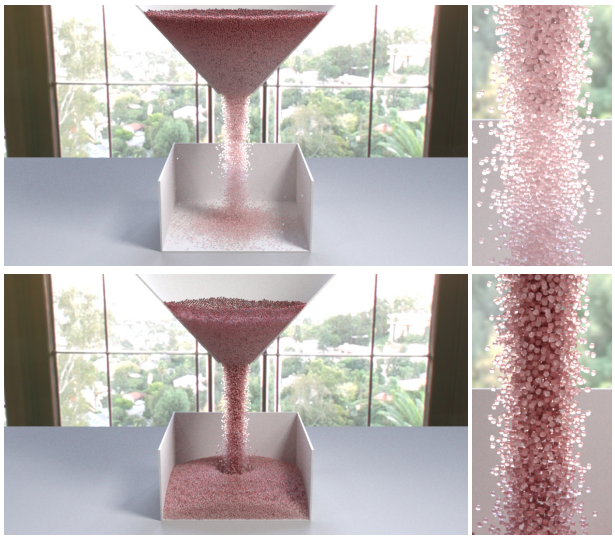


Figure 10: Renderings of explicitly stored grains. This example shows two frames of an animation in which translucent grains with spatially varying optical densities going through a funnel. Please refer to the accompanying video for the full animation.

model used in Figure 1 where the ring consists of 100 billion grains with two main types: bright ice blocks and dark space rocks. Grains within each type have continuously varying sizes and albedo, and the spatial distribution and optical properties of the grains fully drive the macro-scale color variation on the ring. Granular materials with this level of scale and complexity cannot be handled easily with prior methods. Our technique offers the generality to efficiently model and render this material, and our multi-scale renderings closely match the reference at vastly varying physical scales. Please see the supplemental video for an animated version of this result.

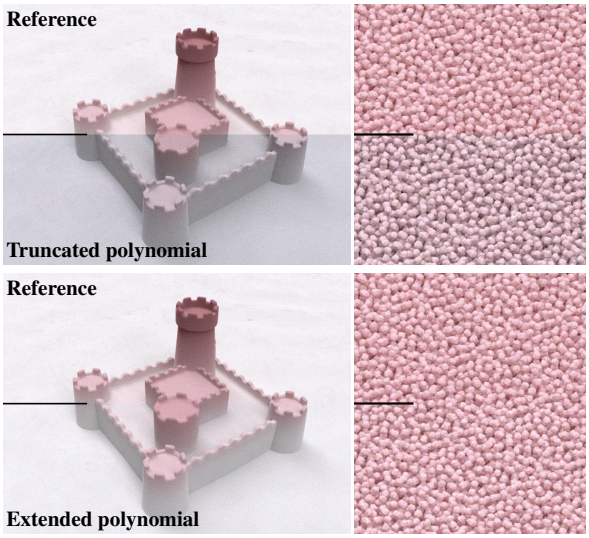


Figure 11: Renderings of our tile-based representation. This castle model consists of 4.5 billion highly scattering grains with albedo up to 0.9995 and smoothly changing colors. With our extended polynomial representation for the GSDF and VPT parameters, the result matches the reference closely. When using conventional truncated polynomials, the result suffers from high energy loss.

6. Discussion and Conclusion

Limitations and future work. Our symbolic and differentiable precomputation only supports varying grain optical properties at render time. Allowing grain shapes to change by differentiating with respect to its geometry may be a useful extension. Also, integrating efficient approximated rendering methods such as precomputed shell transport (e.g., [MWM07]) and diffusion methods (e.g., [AWB11]) into our multi-scale rendering framework can be valuable. Lastly, grains described using our tile-based method cannot be easily animated as the grain locations are largely fixed at the tile level. An interesting challenge is to generalize the template tile to support animations in a temporally consistent fashion.

Conclusion. We introduced a new technique to render heterogeneous granular materials. Leveraging a symbolic and differentiable Monte Carlo process, our technique allows a single precomputation of grain-light interaction to be reused for grains with varying optical properties. This flexibility enables efficient multi-scale rendering of materials with continuously changing grain properties. Further, we show how highly scattering grains can be supported efficiently by extending the polynomials analytically. In addition, we presented a tile-based framework to model large-scale granular materials. Our approach groups individual grains into tiles that are realized on the fly based on a precomputed template tile. Different tiles can have varying grain counts and sizes, allowing grains distributed in complex and non-uniform ways to be described compactly.

We demonstrated the generality and effectiveness of our technique on a few examples, featuring heterogeneous granular materials with varying scales, grain distributions and optical properties.

Acknowledgments

We thank the anonymous reviewers for their suggestions and comments. This work was supported by NSF grant 1813553.

References

[AWB11] ARBREE A., WALTER B., BALA K.: Heterogeneous subsurface scattering using the finite element method. *IEEE Transactions on Visualization and Computer Graphics 17*, 7 (2011), 956–969. 7, 12

[BNH*16] BLUMER A., NOVÁK J., HABEL R., NOWROUZEZAHRAI D., JAROSZ W.: Reduced aggregate scattering operators for path tracing. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 461–473. 3

[BRM*18] BITTERLI B., RAVICHANDRAN S., MÜLLER T., WRENNINGE M., NOVÁK J., MARSCHNER S., JAROSZ W.: A radiative transfer framework for non-exponential media. *ACM Trans. Graph. 37*, 6 (2018), 225:1–225:17. 3

[Dul12] DULLIEN F. A.: *Porous media: fluid transport and pore structure*. Academic press, 2012. 4

[DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graph. 35*, 1 (2015), 9:1–9:13. 2

[GCH*19] GUO J., CHEN Y., HU B., YAN L.-Q., GUO Y., LIU Y.: Fractional gaussian fields for modeling and rendering of spatially-correlated media. *ACM Trans. Graph. 38*, 4 (2019), 45:1–45:13. 3

[GLZ16] GKIOULEKAS I., LEVIN A., ZICKLER T.: An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision* (2016), Springer, pp. 685–701. 3

[GZB*13] GKIOULEKAS I., ZHAO S., BALA K., ZICKLER T., LEVIN A.: Inverse volume rendering with material dictionaries. *ACM Trans. Graph. 32*, 6 (2013), 162:1–162:13. 3

[HR13] HAŠAN M., RAMAMOORTHI R.: Interactive albedo editing in path-traced volumetric materials. *ACM Trans. Graph. 32*, 2 (2013), 11:1–11:11. 3, 5

[JAG18] JARABO A., ALIAGA C., GUTIERREZ D.: A radiative transfer framework for spatially-correlated materials. *ACM Trans. Graph. 37*, 4 (2018), 83:1–83:13. 3

[Jak18] JAKOB W.: Mitsuba renderer, 2018. http://www.mitsuba-renderer.org. 10

[JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHI R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Trans. Graph. 33*, 4 (2014), 115:1–115:10. 2

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Trans. Graph. 25*, 3 (2006), 509–518. 8

[Kra03] KRAAIJPOEL D. A.: *Seismic ray fields and ray field maps: theory and algorithms*. PhD thesis, 2003. 5

[KSZ*15] KHUNGURN P., SCHROEDER D., ZHAO S., BALA K., MARSCHNER S.: Matching real fabrics with micro-appearance models. *ACM Trans. Graph. 35*, 1 (2015), 1:1–1:26. 3

[LADL18] LI T.-M., AITTALA M., DURAND F., LEHTINEN J.: Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. 37*, 6 (2018), 222:1–222:11. 3

[LDHM16] LIU A. J., DONG Z., HAŠAN M., MARSCHNER S.: Simulating the structure and texture of solid wood. *ACM Trans. Graph. 35*, 6 (2016), 170:1–170:11. 2

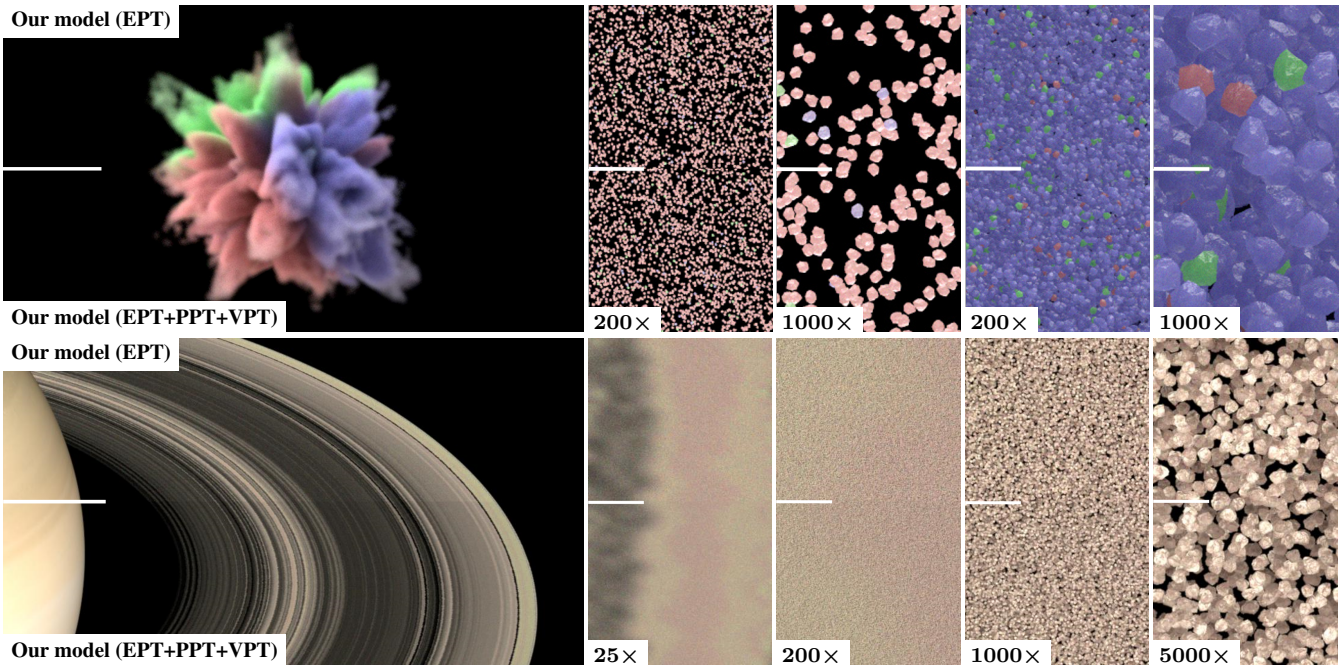


Figure 12: Large-scale models represented with our tile-based approach. **(top)** An explosion scene involving 10 billions multi-colored grains with smoothly varying colors and sizes. **(bottom)** A virtual scene modeled after planet Saturn with the ring comprised of 100 billion ice and rock grains.

[LZB17] LUAN F., ZHAO S., BALA K.: Fiber-level on-the-fly procedural textiles. *Comput. Graph. Forum 36*, 4 (2017), 123–135. 2

[MPG*16] MÜLLER T., PAPAS M., GROSS M., JAROSZ W., NOVÁK J.: Efficient rendering of heterogeneous polydisperse granular media. *ACM Trans. Graph. 35*, 6 (2016), 168:1–168:14. 2, 3, 6, 7, 8, 10

[MPH*15] MENG J., PAPAS M., HABEL R., DACHSBACHER C., MARSCHNER S., GROSS M., JAROSZ W.: Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph. 34*, 4 (2015), 49:1–49:13. 2, 3, 4, 6, 7, 8, 10, 11

[MWM07] MOON J. T., WALTER B., MARSCHNER S. R.: Rendering discrete random media using precomputed scattering solutions. In *Proceedings of the 18th Eurographics conference on Rendering Techniques* (2007), Eurographics Association, pp. 231–242. 2, 7, 12

[SDST06] SKOGE M., DONEV A., STILLINGER F. H., TORQUATO S.: Packing hyperspheres in high-dimensional euclidean spaces. *Physical Review E 74*, 4 (2006), 041127:1–041127:11. 8

[SZK15] SCHRÖDER K., ZINKE A., KLEIN R.: Image-based reverse engineering and visual prototyping of woven cloth. *IEEE transactions on visualization and computer graphics 21*, 2 (2015), 188–200. 2

[WMHL65] WOODCOCK E., MURPHY T., HEMMINGS P., LONGWORTH S.: Techniques used in the gem code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems* (1965), vol. 557. 6

[Yuk15] YUKSEL C.: Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 25–32. 8

[ZJMB11] ZHAO S., JAKOB W., MARSCHNER S., BALA K.: Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph. 30*, 4 (2011), 44:1–44:10. 2

[ZLB16] ZHAO S., LUAN F., BALA K.: Fitting procedural yarn models for realistic cloth rendering. *ACM Trans. Graph. 35*, 4 (2016), 51:1–51:11. 2

[ZWDR16] ZHAO S., WU L., DURAND F., RAMAMOORTHI R.: Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph. 35*, 6 (2016), 166:1–166:11. 3