Efficient Scene Appearance Aggregation for Level-of-Detail Rendering

YANG ZHOU, University of California, Santa Barbara, USA
TAO HUANG, University of California, Santa Barbara, USA
RAVI RAMAMOORTHI, University of California, San Diego, USA
PRADEEP SEN, University of California, Santa Barbara, USA
LING-QI YAN, University of California, Santa Barbara, USA

Fig. 1. **Left**: *Metropolis*, a cityscape rendered with our scene aggregation approach. The scene includes 82 unique buildings and 270 instances and originally requires 46.9 GB to store, making it challenging to render while stay in in-core memory. Our representation drastically reduces the size to 5.33 GB while preserving the detailed appearance. **Right Top**: Each instance selects the appropriate LoD resolution where the projected voxel size matches the pixel footprint (rounded to the nearest power of two). **Right Bottom**: As a result, close-view instances are rendered with finer voxels while distant instances are rendered with coarser voxels.

Creating an appearance-preserving level-of-detail (LoD) representation for arbitrary 3D scenes is a challenging problem. The appearance of a scene is an intricate combination of both geometry and material models, and is further complicated by correlation due to the spatial configuration of scene elements. We present a novel volumetric representation for the aggregated appearance of complex scenes and an efficient pipeline for LoD generation and rendering. The core of our representation is the *Aggregated Bidirectional Scattering Distribution Function* (ABSDF) that summarizes the far-field appearance of all surfaces inside a voxel. We propose a closed-form factorization of the ABSDF that accounts for spatially varying and orientation-varying material parameters. We tackle the challenge of capturing the correlation existing locally within a voxel and globally across different parts of the scene. Our method faithfully reproduces appearance and achieves higher quality than existing scene filtering methods while being inherently efficient to render. The memory footprint and rendering cost of our representation are independent of the original scene complexity.

CCS Concepts: • **Computing methodologies** → **Reflectance modeling**; **Visibility**.

Additional Key Words and Phrases: level-of-detail, aggregation, prefiltering, appearance modeling

---

Authors' Contact Information: Yang Zhou, yzhou426@cs.ucsb.edu, University of California, Santa Barbara, Santa Barbara, California, USA; Tao Huang, tao_huang@ucsb.edu, University of California, Santa Barbara, Santa Barbara, California, USA; Ravi Ramamoorthi, ravir@cs.ucsd.edu, University of California, San Diego, La Jolla, California, USA; Pradeep Sen, psen@ece.ucsb.edu, University of California, Santa Barbara, Santa Barbara, California, USA; Ling-Qi Yan, lingqi@cs.ucsb.edu, University of California, Santa Barbara, Santa Barbara, California, USA.

1 Introduction

Modern physically based rendering is widely adopted to synthesize photorealistic images, animations, and immersive 3D experiences. Generating content at such level of realism requires large-scale assets with extremely detailed geometry, textures, and sophisticated material models. This presents significant challenges to the rendering process both in terms of storage and speed. Among them, one prominent issue comes from the mismatch between scene complexity and image resolution. In an open-world environment, it is typical to only have a small portion of the scene contribute to the foreground, while the majority of the scene is minified in the background. It is wasteful to load and render the entirety of the scene when the image resolution is not even enough to resolve the details. Moreover, the rendering cost of different pixels can be highly uneven. Some pixels may cover a drastically more complex part of the scene than others and thus require an excessive sampling budget for convergence. On the other hand, ignoring such complexity often leads to aliasing, artifacts or incorrect appearance.

Level-of-detail (LoD) techniques reduce the heavy, unbalanced rendering cost by converting, or prefiltering, the original scene to a multi-scale representation in a precomputation step. Depending on how much detail is required for each pixel, only an appropriate scale of the representation is accessed and used for rendering. In this way, LoD techniques are able to decouple rendering cost from the original scene complexity and distribute the cost evenly among pixels.

In order to improve efficiency, LoD techniques usually perform simplification to the original geometry. A key challenge for any LoD technique is that it should preserve the original appearance after the simplification. When viewed at distance, the appearance is a compound phenomenon of both geometry and material models. Simply discarding or averaging geometry would result in appearance mismatch and artifacts [Luebke et al. 2003]. Instead, the technique should condense the effect of the original geometry into the simplified representation. This process can be called appearance aggregation. It is important to realize that the aggregated appearance can be more complex than, say, the original material models because it describes more information. However, it can still be advantageous performance-wise compared to tracing the explicit geometry.

Many existing LoD solutions converts geometry to volumes for filtering or downsampling. The recurring difficulty for these solutions is the loss of geometric correlation. We lose track of how the geometry is distributed locally within a volume when it is abstracted away. Furthermore, if multiple regions are simplified separately, we lose track of the long-range visibility caused by the specific spatial configuration between geometry of different regions. Correlation exists ubiquitously in different types of scenes, such as those containing large, connected surface or regularly organized structures. Ignoring correlation leads to incorrect appearance for the LoD representation.

In this work, we propose an efficient volumetric scene appearance aggregation method for LoD rendering. Our representation supports arbitrary types of scenes geometry from completely opaque surfaces to stochastically distributed structures, and a wide range of appearance from glossy to diffuse. At the heart of our representation is the Aggregated Bidirectional Scattering Distribution Function (ABSDF) that summarizes the appearance of all surfaces inside a voxel. Contrary to existing volume-based methods, our method inherently keeps track of long-range correlation by recording the global visibility originated from a voxel and from the scene boundary. Simultaneously, we propose a novel truncated ellipsoid primitive to better handle the local correlation within a voxel. We focus on the appearance of a scene at far field, as is the case when an LoD representation gains the most benefit. Similar to Bako et al. [2023], we focus on the appearance with direct illumination, which is arguably the more challenging part compared to the indirectly illuminated counterpart as it is subject to more visible artifacts such as leaking and bloating. Our method achieves high rendering fidelity by preserving the complex visual appearance caused by both geometry and materials (Fig. 1, Fig. 15, and Fig. 16).

To summarize, our contributions include:
- A novel formulation for representing and rendering far-field scene aggregates for arbitrary scenes with the *Aggregated Bidirectional Scattering Distribution Function* (ABSDF).
- A closed-form factorization of the aggregated appearance that captures all-frequency and view-dependent effects. The resulting model supports efficient evaluation and importance sampling.
- A practical solution that handles local correlation by truncated ellipsoid primitives and long-range correlation by recording global visibility.

- An efficient scene aggregation pipeline that is scalable to large, complex assets and offers asymptotic memory saving and rendering speed boost.

2 Related work

Representing scenes and appearance at multiple scales to improve rendering efficiency and quality is a long-standing problem in computer graphics. We draw inspiration from various previous work ranging from surface-based approaches to volume-based approaches, together with hybrid approaches in between. In addition, the recent advances of neural representations provide a set of new tools proven to be effective in certain graphics applications.

Mesh simplification. Polygon meshes are by far the most common representation of 3D models in computer graphics. A large amount of study has been focused on algorithms that simplify a complex mesh by collapsing edges and merging vertices [Hoppe 1996; Garland and Heckbert 1997]. Some attempts have been made to extend mesh simplification to consider appearance to a limited extent [She et al. 2019; Cook et al. 2007]. Mesh simplification techniques are widely employed in movie and video game production [Karis et al. 2021]. However, they are fundamentally unable to preserve the complex appearance that is a combination of both detailed geometry and material models. More recently, Hasselgren et al. [2021] jointly optimize triangle meshes and material parameters to minimize the image-space difference to the target scene by a differentiable rasterizer. However, the shading model is limited to be the same before and after optimization. The optimization also ignores global effects such as shadows.

Surface appearance filtering. Surface-based filtering techniques focus on filtering the spatially-varying material attributes and microscale geometric details while keeping the original macro-scale surface geometry. Normal map filtering, for example, converts the normal directions inside a footprint to a normal distribution function (NDF) to preserve highlights when viewed from afar [Toksvig 2005; Han et al. 2007; Olano and Baker 2010; Kaplanyan et al. 2016]. Xu et al. propose to jointly mipmap BRDF and normal maps [Xu et al. 2017]. Glints rendering [Yan et al. 2014, 2016] focuses on resolving the highlight from specular micro-geometry, which is essentially the same problem. However, both the spatial resolution of the normal maps and the angular resolution of the NDFs are much higher. The source normal maps can also be procedurally generated to alleviate the high memory cost [Jakob et al. 2014; Zirr and Kaplanyan 2016; Wang et al. 2020]. Displacement map filtering incorporates the microscale geometric details provided by displacement maps inside a footprint into a shading model [Dupuy et al. 2013; Wu et al. 2019]. Bi-scale material design models the macro-scale appearance of an object by designing its microscale details and aggregates their appearance [Wu et al. 2011; Iwasaki et al. 2012]. Bidirectional texture functions (BTFs) represent non-parametric 6D spatially-varying surface appearance. Filtering BTFs offers significant memory savings and a performance boost [Jarabo et al. 2014]. Surface-based techniques successfully simplify microscale details by prefiltering them into an appearance model. However, they do not alter macro-scale

geometry, thus they are not helpful when macro-scale geometry is the dominant factor in scene complexity.

Volumetric appearance models and filtering. Using volumes to represent complex geometry has been explored extensively since first introduced by Kajiya and Kay [1989]. Volumes are traditionally used to accelerate the rendering of dense, unstructured geometry such as fur, hair, and foliage [Neyret 1998; Moon et al. 2008]. Jakob et al. [2010] proposes the microflake theory that extends the radiative transfer equation (RTE) [Chandrasekhar 1960] to anisotropic participating media, enabling volumes to represent a wider range of appearance such as fabric and cloth [Zhao et al. 2011, 2012]. Heitz et al. [2015] further proposes the SGGX distribution to construct efficient microflake phase functions that support linear interpolation and closed-form importance sampling. As a high-resolution volume can be very memory-intensive, several works consider the problem of downsampling microflake volumes while preserving the important self-shadowing effect [Zhao et al. 2016; Loubet and Neyret 2018]. The classic volumetric light transport theory that builds on the RTE assumes independently distributed scatterers and thus does not support spatial correlation, limiting its expressiveness for general scene representation. More recently, it has been further extended to support spatially-correlated participating media through different formulations [Jarabo et al. 2018; Bitterli et al. 2018]. Vicini et al. [2021] proposes an empirical non-exponential transmittance model that, while not physically-based, improves the ability to model correlation and opaque surfaces when combined with data-driven optimization. While volume-based techniques are able to simplify macro-scale geometry, volumetric light transport itself is significantly harder to solve than surface light transport and typically takes a longer time to converge for Monte Carlo path tracing.

Another line of works focuses on building efficient voxel-based data structures. Crassin et al. [2009] and Laine and Karras [2010] propose different variants of a sparse voxel octree (SVO) to render massive volumes at interactive rates. The SVO data structure can be further specialized to support even higher resolution [Kämpe et al. 2013]. Building on top of SVO, Heitz and Neyret [2012] proposes a representation to filter the appearance of detailed surfaces with the ability to reproduce view-dependent effects and account for correlation of occlusion and attributes with visibility. However, they only support opaque surfaces modeled by a boundary representation. Thus, their work is not applicable to a wide variety of subjects consisting of dense, unstructured geometry.

Hybrid approaches. A number of works attempt to combine the advantages of surface-based techniques and volume-based techniques. Dupuy et al. [2016] draws a theoretical connection between microfacet and microflake theories. Granular material rendering techniques achieve acceleration by switching representation at different scales of light transport [Moon et al. 2007; Meng et al. 2015; Müller et al. 2016; Zhang and Zhao 2020]. Grains are only explicitly traced during initial bounces. For longer-scale light transport and multiple scattering, grains are replaced with a volumetric representation that is rendered by volumetric path tracing and eventually diffusion methods. Loubet and Neyret [2017] proposes a hybrid LoD technique that performs a binary classification on the input

scene to divide it into a mesh part and a volume part at each scale. Subsequently, the mesh part undergoes mesh simplification and the volume part is represented by a microflake participating medium. While the idea sounds straightforward, the classification unfortunately suffers from ambiguity and the technique produces artifacts when misclassification happens. Additionally, mesh simplification may drastically alter surface curvature that results in incorrect glossy appearance, as shown in Fig. 13.

Neural representations. Neural implicit representations are shown to be particularly effective at compactly reconstructing signals in low-dimensional spaces such as radiance fields and shapes [Mildenhall et al. 2020; Lombardi et al. 2021; Martel et al. 2021; Müller et al. 2022; Park et al. 2019]. While most works focus on point-wise query and inference, some techniques build multi-scale representations that support range queries for anti-aliasing [Barron et al. 2021; Takikawa et al. 2021]. However, most neural implicit representations are unable to model full appearance, with limited capability for relighting [Bi et al. 2020; Lyu et al. 2022; Baatz et al. 2022]. For a more comprehensive review, we refer readers to two recent surveys on the subject [Tewari et al. 2022; Xie et al. 2022].

Traditional surface-based techniques can be enhanced by neural components. Kuznetsov et al. [2021, 2022] achieve BTF compression and filtering by simultaneously training a latent texture pyramid and a small multilayer perceptron (MLP) decoder that supports isotropic range queries. Gauthier et al. [2022] improve normal map filtering by using a MLP cascade to learn downsampling kernels.

Recently, Bako et al. [2023] propose a deep learning based appearance-prefiltering framework. An input scene is converted to a volumetric representation where each voxel records a monochromatic phase function, an average albedo, and a 4D view-dependent coverage mask. To reduce the otherwise infeasible memory requirement, each type of data is compressed by a separate encoder-decoder network that produces per-voxel latent vectors. The volume is rendered by a beam tracer that traverses the voxels and decodes them for shading and transmittance computation. The method preserves accurate appearance but at a heavy cost in both precomputation and rendering. The typical precomputation time is reported to be 0.5 to 2 days on a GPU cluster with 256 NVIDIA Volta GPUs. The compressed per-voxel size is still large with 256 floats. In addition, the beam tracer must traverse voxels ordered by distance to correctly compute transmittance by accumulating the coverage masks from each voxel. In contrast, our method only requires a much lighter precomputation pass, a smaller memory cost, offers much faster rendering speed, and results in similar rendering quality. Weier et al. [2023] propose a neural prefiltering pipeline by learning a compressed representation of the intra-voxel light transport. Two independent networks for appearance and visibility are trained with a multi-level feature grid. The method handles diffuse-like appearance well and supports indirect lighting. However, it struggles at preserving glossy appearance and capturing all-frequency directional signals.

3 Far-field Appearance Aggregation and Factorization

3.1 Overview

Our goal is to develop an appearance-preserving representation of a scene that is independent of the original geometry complexity, which we term *scene aggregate*. Fig. 2 provides an overview of our method. When measured externally, the general light transport of a scene aggregate can be characterized as an 8D function of incident/outgoing positions (on a suitable bound of the scene) and directions, similar to a BSSRDF. However, directly computing and storing such a function is impractical due to the prohibitive memory requirement. It is also not necessary as one might as well simply switch back to the original representation for near-field appearance. Therefore, our formulation is based on the far-field assumption. When a scene is sufficiently far from the measuring sensor and emitters that the sensor can no longer distinguish the internal spatial structure, we may drop the positional dependency by integrating the 8D light transport function over positions. The resulting 4D function of incident/outgoing directions describes the far-field appearance of the scene aggregate and we name it *Aggregated Bidirectional Scattering Distribution Function* (ABSDF), again due to its similarity to a BSDF. In §3, §4, and §5, we define the ABSDF and present an efficient, closed-from factorization of it.

In practice, an entire scene is usually too large to be considered far-field all together. We apply spatial subdivision to the scene at a suitable resolution given a certain pixel footprint such that the subset of the scene included in each voxel satisfies the far-field assumption. This introduces the subsequent problem of accumulating the outgoing radiance from voxels and eventually measuring the pixel intensities. Crucially, the accumulation problem is non-trivial because the spatial configuration of voxels is not independent. Traditional volumetric representations model a scene as independently distributed particles, which is incorrect because a scene made of surfaces typically exhibits spatial correlation and ignoring such correlation leads to artifacts or inaccurate appearance. In §6, we analyze the problem of spatial correlation in detail, discuss our strategies to preserve correlation, and derive the formulation for voxel accumulation. We provide a summary for commonly used symbols throughout the paper in Table 1.

3.2 Defining ABSDF

We consider a subset of a scene A that consists of a set of surfaces. From a point $x \in A$, the outgoing radiance given some direct incident radiance $L_i(x, \omega_i)$ is calculated by the following equation [Cohen and Wallace 1993]:

$$L_o(x, \omega_o) = \int_{\mathbb{S}^2} f(x, \omega_i, \omega_o) L_i(x, \omega_i) \langle n_x \cdot \omega_i \rangle V(x, \omega_i) \, d\omega_i, \quad (1)$$

where $f(x, \omega_i, \omega_o)$ is the surface BRDF at x, and we explicitly write the visibility term $V(x, \omega_i)$. We are interested in the average outgoing radiance from A when viewed from ω_o, which can be written as a weighted average of per-point outgoing radiance masked by

Table 1. Table of notation.

Symbol	Explanation	Def.
A	A set of surfaces (in a voxel)	
$|A|$	Surface area of A	
$|A|_\omega$	Projected surface area of A along ω	§3.2, Eq. 2
$\hat{f}$	Aggregated Bidirectional Scattering Distribution Function (ABSDF)	§3.2, Eq. 3
$\hat{f}_{\text{novis}}$	ABSDF without visibility	§3.3, Eq. 7
n	Surface normal	
$p_N(n)$	Surface normal distribution function	§3.3, Eq. 8
$D_{\text{sggx}}(n)$	SGGX distribution parameterized by eigenbasis R and roughness $\boldsymbol{\alpha} := (\alpha_x, \alpha_y)$	§3.3
β	Material parameters at point x, including roughness α, basecolor β^c (spectral), metallic β^m (scalar), and specular intensity β^s (scalar)	§3.3
γ	Concatenation of n and β	§3.3
$p_Y(\gamma)$	Joint distribution of γ	§3.3
B	Truncated ellipsoid primitive	§6.1
$c(\omega)$	Primitive coverage	§6.1, Eq. 24
$\hat{V}$	Aggregated interior visibility	§6.2, Eq. 25
$\hat{V}_b$	Aggregated boundary visibility	§6.2, Eq. 26
$\langle - \cdot - \rangle$	Clamped dot product	

another visibility term along ω_o:

$$L_o(\omega_o) = \frac{1}{|A|_{\omega_o}} \int_A L_o(x, \omega_o) \langle n_x \cdot \omega_o \rangle V(x, \omega_o) \, dx,$$

$$|A|_{\omega_o} = \int_A \langle n_x \cdot \omega_o \rangle \, dx, \quad (2)$$

where $|A|_{\omega_o}$ is the projected area of A along ω_o. We can apply the far-field assumption such that the incident radiance is independent of positions $L_i(x, \omega_i) \approx L_i(\omega_i)$ and rearrange Eq. 2 by reordering the integrations:

$$L_o(\omega_o) = \int_{\mathbb{S}^2} \hat{f}(\omega_i, \omega_o) L_i(\omega_i) \, d\omega_i,$$

$$\hat{f}(\omega_i, \omega_o) = \frac{1}{|A|_{\omega_o}} \int_A \Big(f(x, \omega_i, \omega_o) \langle n_x \cdot \omega_i \rangle \langle n_x \cdot \omega_o \rangle \quad (3)$$

$$V(x, \omega_i) V(x, \omega_o) \Big) \, dx.$$

We define $\hat{f}(\omega_i, \omega_o)$ as the ABSDF of A as it captures the intrinsic geometrical and material characteristics of A and is independent of external sensors or emitters. It is not hard to see that the AB-SDF satisfies energy conservation as long as $f(x, \omega_i, \omega_o)$ is energy-conserving. The ABSDF can be interpreted as an extension of the effective BRDF [Wu et al. 2011] where surfaces are not longer confined to a macrosurface or a heightfield, but allowed to be arranged arbitrarily in free space. It also satisfies a generalized form of reciprocity that is similar to the situation in the microflake theory [Jakob
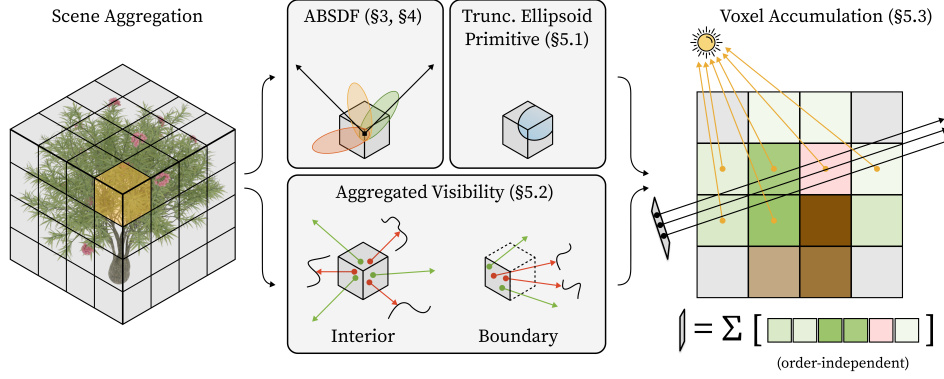
Fig. 2. An overview of our method. We start by voxelizing a scene such that the voxel size matches the given pixel footprint. For each voxel, we model its aggregated appearance by its ABSDF. To preserve the local spatial correlation, we use a truncated ellipsoid primitive that describes the intra-voxel geometric distribution. To preserve the long-range correlation, we record global aggregated visibility. Both lead to accurate voxel accumulation that is order-independent.

et al. 2010]. To summarize:

$$\int_{\mathbb{S}^2} \hat{f}(\omega_i, \omega_o) \, d\omega_i \leq 1 \tag{4}$$

$$|A|_{\omega_o} \hat{f}(\omega_i, \omega_o) = |A|_{\omega_i} \hat{f}(\omega_o, \omega_i) \tag{5}$$

We note that it is possible to use the visible projected area $|A_v(\omega_o)| = \int_A \langle n_x \cdot \omega_o \rangle V(x, \omega_o) \, dx$ as the normalization term in Eq. 3. In fact, the choice is not critical as the term will eventually be cancelled out (Eq. 24, Eq. 28). We choose to use $|A|_{\omega_o}$ for simplicity. In practice, neither $|A_v(\omega_o)|$ nor $|A|_{\omega_o}$ needs to be computed or stored.

3.3 A Closed-form Factorization of the ABSDF

According to Eq. 3, an ABSDF is defined by integrating the product of base material, foreshortening factors, and bidirectional visibility over the underlying surfaces. This is challenging, as in general, no closed-form solution exists. However, we would also like to avoid stochastic evaluation or numerical integration which would greatly increase the rendering cost and undermine the purpose of scene aggregation. To achieve closed-form evaluation and importance sampling, we factor the ABSDF with the following steps.

Separate Visibility. We perform a splitting approximation that separates the integration of the bidirectional visibility from the rest:

$$\hat{f}(\omega_i, \omega_o) \approx \frac{1}{|A|_{\omega_o}} \int_A f(x, \omega_i, \omega_o) \langle n_x \cdot \omega_i \rangle \langle n_x \cdot \omega_o \rangle \, dx$$
$$\frac{1}{|A|} \int_A V(x, \omega_i) V(x, \omega_o) \, dx. \tag{6}$$

This is similar to the approximation by Jiménez et al. [2016]. We focus on the first integral in the rest of this section and further describe how to incorporate visibility into our framework in §6.2.

The distribution form of the ABSDF. With the visibility terms separated, we convert the ABSDF into a convolution between the base material and the joint distribution of material parameters. Similar operations have been employed in normal map and displacement map filtering techniques [Han et al. 2007; Olano and Baker 2010; Dupuy et al. 2013], but they usually only consider the distribution

of surface normals. Our formulation can be seen as a generalization that incorporates all spatially-varying parameters. Let $\gamma_x := (n_x, \beta_x)$ be a vector consisting of surface normal and all material parameters at x. It can also be interpreted as a value of a random vector Y with joint PDF $p_Y(\gamma)$. We can rewrite the ABSDF (no visibility) as follows:

$$\hat{f}_{\text{novis}}(\omega_i, \omega_o) = \frac{1}{|A|_{\omega_o}} \int_A f(\omega_i, \omega_o; \gamma_x) \langle n_x \cdot \omega_i \rangle \langle n_x \cdot \omega_o \rangle \, dx$$
$$= \frac{|A|}{|A|_{\omega_o}} \int_\Gamma f(\omega_i, \omega_o; \gamma) \langle n \cdot \omega_i \rangle \langle n \cdot \omega_o \rangle p_Y(\gamma) \, d\gamma, \tag{7}$$

where Γ is the product space of all parameters.

Surface normal distribution function. The marginal distribution of surface normals $p_N(n)$, or the surface NDF, is important as it affects the glossiness and anisotropy of the aggregated appearance. It can also be complex because the underlying surfaces may be arbitrarily oriented. We use a mixture of the SGGX distribution [Heitz et al. 2015] as a compact yet expressive representation for the surface NDF:

$$p_N(n) = \sum_{i=1}^{k} w_i D_{\text{sggx}}^i(n), \tag{8}$$

where the weights $\{w_i\}$ are positive and sum to 1. We describe the fitting process for the mixture model in §7.1.

As will be seen in §4 and §5.2, ours factorization involves convolving $p_N(n)$ with another isotropic spherical distribution $g(\omega; n)$. Because $p_N(n)$ is a mixture of SGGX lobes, the result is the sum of the convolution between each lobe $D_{\text{sggx}}^i(n)$ and $g(\omega; n)$. We propose to represent the per-lobe convolution as a similar but roughened SGGX. We first parameterize an SGGX distribution by its eigenbasis $R := (\omega_1, \omega_2, \omega_3)$ and anisotropic roughness $\boldsymbol{\alpha} := (\alpha_x, \alpha_y)$. The parameterization is detailed in the supplemental document. The convolution can then be written as

$$\int_{\mathbb{S}^2} g(\omega; n) D_{\text{sggx}}(n; R, \boldsymbol{\alpha}) \, dn \approx D_{\text{sggx}}(\omega; R, \boldsymbol{\alpha}_+), \tag{9}$$

where R stays fixed but $\boldsymbol{\alpha}$ gains additional values. This is inspired by Xu et al. [2013], where a similar approximation is made for

anisotropic Spherical Gaussians. While Xu et al. seeks a symbolic approximation, we simply perform a nonlinear least-square fit to find the best mapping $\alpha_+ = M(\alpha, g)$. Note that this mapping is *scene-independent* and typically smooth, thus only requiring pre-comptation once and negligible storage. We observe accurate fits for all our target distributions g. We provide derivation details and numerical validation with different g in the supplemental document. Following Eq. 9, the post-convolution distribution for the entire $p_N(n)$ is

$$D_{\text{conv}}(\omega) = \int_{\mathbb{S}^2} g(\omega; n) p_N(n) \, \mathrm{d}n \approx \sum_{i=i}^{k} w_i D_{\text{sggx}}^i(\omega; R^i, \alpha_+^i). \quad (10)$$

Base material. The ABSDF is dependent on the underlying surface base materials. For the widest applicability, it is desirable to support material models used by existing assets. Therefore, we target the Disney Principled BRDF [Burley 2012], which is one of the most commonly used models in production and capable of recreating a wide range of appearance. The Disney BRDF is a sophisticated model consisting of multiple lobes. We preserve its core feature but make three modifications to the original model:

(1) For diffuse reflection, we use the simpler Lambertian model instead of the original empirical model with retro-reflection.

(2) We omit the optional sheen and clearcoat lobes.

(3) We assume surfaces are double-sided.
The modified model can be written as

$$f_{\text{disney}}(\omega_i, \omega_o) = f_d(\omega_i, \omega_o) + f_s(\omega_i, \omega_o),$$
$$f_d(\omega_i, \omega_o) = \frac{1}{\pi}(1 - \beta^m)\beta^c,$$
$$f_s(\omega_i, \omega_o) = \frac{D(\omega_h; \alpha)G(\omega_i, \omega_o; \alpha)}{4|n \cdot \omega_o||n \cdot \omega_i|}\Big(\beta^m F(\omega_h, \omega_o; \beta^c) + \quad (11)$$
$$(1 - \beta^m)F(\omega_h, \omega_o; \beta^s)\Big),$$

where $f_s(\omega_i, \omega_o)$ is the specular component that consists of both metallic and dielectric Fresnel reflection, $f_d(\omega_i, \omega_o)$ is the diffuse component, $D(\omega_h)$ is the Trowbridge-Reitz (GGX) distribution as the microfacet distribution, $G(\omega_i, \omega_o)$ is the shadowing-masking function, and $F(\omega_h, \omega_o; r_0)$ is the Schlick Fresnel reflectance:

$$F(\omega_h, \omega_o; r_0) = r_0(1 - \mathcal{F}_c) + \mathcal{F}_c,$$
$$\mathcal{F}_c = (1 - |\omega_h \cdot \omega_o|)^5,$$

where r_0 is the normal incidence reflectance (either β^c for the metallic lobe or β^s for the dielectric lobe). The model is controlled by a set of parameters $\beta \coloneqq (\alpha, \beta^c, \beta^m, \beta^s) = $ (roughness, basecolor, metallic, specular intensity), which can all be spatially-varying. We now focus on factoring of each component, starting from the simpler diffuse component.

4 Diffuse ABSDF Factorization

We substitute $f_d(\omega_i, \omega_o)$ into the integral of Eq. 7 and perform a split by first assuming β^c and β^m are orientation-independent. This means the joint parameter PDF $p_Y(\gamma)$ becomes a product of two

marginal PDFs $p_Y(\gamma) \approx p(\beta^c, \beta^m) p_N(n)$ and we have

$$\int_\Gamma f_d(\omega_i, \omega_o; \gamma)\langle n \cdot \omega_i \rangle \langle n \cdot \omega_o \rangle p_Y(\gamma) \, \mathrm{d}\gamma \approx \quad (12)$$
$$\frac{1}{\pi} \int_{[0,1]^2} (1 - \beta^m)\beta^c p(\beta^c, \beta^m) \, \mathrm{d}\beta^c \mathrm{d}\beta^m \int_{\mathbb{S}^2} \langle n \cdot \omega_i \rangle \langle n \cdot \omega_o \rangle p_N(n) \, \mathrm{d}n.$$

We extend our formulation to handle orientation-varying material parameters in §5.4. After the splitting, the left integral can be simply represented by the means and second-order moments of the parameters

$$\int_{[0,1]^2} (1 - \beta^m)\beta^c p(\beta^c, \beta^m) \, \mathrm{d}\beta^c \mathrm{d}\beta^m = \mathbb{E}[\beta^c] - \mathbb{E}[\beta^m \beta^c]. \quad (13)$$

For the right integral, we follow Wang et al. [2009] and fit the clamped dot product function by a Spherical Gaussian (SG). Because SGs are closed under multiplication with a closed-form expression [Wang et al. 2009], we can expand the right integral of Eq. 12 as

$$\int_{\mathbb{S}^2} \langle n \cdot \omega_i \rangle \langle n \cdot \omega_o \rangle p_N(n) \, \mathrm{d}n \approx \int_{\mathbb{S}^2} c \cdot \text{SG}(\omega_h; n, \kappa) p_N(n) \, \mathrm{d}n, \quad (14)$$

where c is the amplitude and κ is the concentration for the product SG. We refer readers to Wang et al. [2009] for the full expressions for them. Notably, the product SG becomes a function of the half vector ω_h. The problem is then reduced to the convolution between an SG and an SGGX, which we solve by employing the convolution technique described in Eq. 10. This comes with a table $\alpha_+ = M_1(\alpha, \kappa)$. Eq. 14 then can be evaluated in closed form given the surface NDF $p_N(n)$.

5 Specular ABSDF Factorization

Next, we describe how to factorize the more challenging specular component of the ABSDF. Together with the diffuse component, our complete factorization will be validated at the end of the section. We start by substituting $f_s(\omega_i, \omega_o)$ into the integral of Eq. 7 and expanding it as

$$\int_\Gamma f_s(\omega_i, \omega_o; \gamma)\langle n \cdot \omega_i \rangle \langle n \cdot \omega_o \rangle p_Y(\gamma) \, \mathrm{d}\gamma =$$
$$\frac{1}{4}\left[(1 - \mathcal{F}_c)\boxed{\int_\Gamma \mathcal{RD}p_Y(\gamma) \, \mathrm{d}\gamma} + \mathcal{F}_c\boxed{\int_\Gamma \mathcal{D}p_Y(\gamma) \, \mathrm{d}\gamma}\right], \quad (15)$$
$$\mathcal{R} = \beta^m \beta^c + (1 - \beta^m)\beta^s,$$
$$\mathcal{D} = D(\omega_h; n, \alpha)G(\omega_i, \omega_o; n, \alpha)\mathbb{1}(n \cdot \omega_o)\mathbb{1}(n \cdot \omega_i),$$

where $\mathbb{1}(\cdot)$ is the Heaviside (step) function that evaluates to 0 or 1. $\mathbb{1}(n \cdot \omega_o)$ and $\mathbb{1}(n \cdot \omega_i)$ appear due to the clamped dot products in Eq. 7. Similar to §4, we split the green highlighted integral in Eq. 15 by assuming $p_Y(\gamma) \approx p_{Y_1}(\beta^c, \beta^m, \beta^s)p_{Y_2}(n, \alpha)$ and extend the formulation to handle orientation-varying material parameters in §5.4:

$$\boxed{\int_\Gamma \mathcal{RD}p_Y(\gamma) \, \mathrm{d}\gamma} \approx \int_{\Gamma_1} \mathcal{R}p_{Y_1}(\gamma) \, \mathrm{d}\gamma \boxed{\int_{\Gamma_2} \mathcal{D}p_{Y_2}(\gamma) \, \mathrm{d}\gamma}, \quad (16)$$

where Γ_1 and Γ_2 are the product space of $(\beta^c, \beta^m, \beta^s)$ and (n, α), respectively. Once again, the left integral can be simply represented

by the means and second-order moments of the parameters

$$\int_{\Gamma_1} \mathcal{R} p_{Y_1}(\gamma) \, d\gamma = \mathbb{E}[\beta^m \beta^c] + \mathbb{E}[\beta^s] - \mathbb{E}[\beta^m \beta^s]. \qquad (17)$$

For the yellow highlighted integral in Eq. 15 and Eq. 16, we propose a closed-form solution with several small, *scene-independent* precomputed tables. The total storage for the tables is less than 5MB in practice. We focus on the characteristic microfacet distribution term D, which we now call D_{mic} in the rest of §5 for better clarity. The shadowing-masking term G is in general very smooth [Ashikhmin et al. 2000; Wang et al. 2009; Kaplanyan et al. 2016]. Our strategy is based on the convolution technique described in §3.3 and include three steps:

- §5.1: Identify the aggregated microfacet distribution $\hat{D}_{\text{mic}}(\omega_h)$.
- §5.2: Convolve $\hat{D}_{\text{mic}}(\omega_h)$ with the surface NDF $p_N(n)$ to get the post-convolution distribution $D_{\text{conv}}(\omega_h)$.
- §5.3: Apply a scaling factor $S(\omega_i, \omega_o)$ to $D_{\text{conv}}(\omega_h)$ to correct leaking beacuse surfaces can be back-facing either to view or lights.

We now proceed to describe each step in detail. The complete model is summarized at end of the section (§5.5, Fig. 3). Additional derivation details and numerical validation are available in the supplemental document.

5.1 Aggregated Microfacet Distribution

As roughness α can vary on the surfaces of A, the microfacet distribution can no longer be represented by one GGX lobe. Let $\mathcal{A}$ be the underlying random variable for the roughness and $p_{\mathcal{A}}(\alpha)$ be its marginal density function. The aggregated microfacet distribution is the expectation of the microfacet distribution

$$\hat{D}_{\text{mic}}(\omega_h) = \int_{[0,1]} D_{\text{mic}}(\omega_h; \alpha) p_{\mathcal{A}}(\alpha) \, d\alpha. \qquad (18)$$

We should represent $p_{\mathcal{A}}(\alpha)$ by a parametric distribution while acknowledging that α is bounded in $[0,1]$. A Gaussian distribution is thus not a valid choice. Instead, we use a beta distribution $\mathcal{B}(\alpha; a, b)$ which has the correct support and is reasonably expressive. The shape parameters a and b can be easily estimated (see supplemental document). Eq. 18 can be interpreted as the weighted average of infinite GGX lobes with different possible α. Since there is no closed-form solution for it, we further propose to approximate $\hat{D}_{\text{mic}}(\omega_h)$ by a weighted average of 2 lobes:

$$\hat{D}_{\text{mic}}(\omega_h) \approx m_1 D_{\text{mic}}(\omega_h; \alpha_1) + m_2 D_{\text{mic}}(\omega_h; \alpha_2), \qquad (19)$$

where $m_1 + m_2 = 1$. The approximation can be extended to use an arbitrary number k of lobes, but we find $k = 2$ provides a good balance between cost and accuracy. We perform a nonlinear least square fit to find the best mapping $(m_1, \alpha_1, \alpha_2) = M_2(a, b)$ given the shape parameters of the beta distribution and store it as a small 2D table.

5.2 Convolution with Surface NDF

The aggregated microfacet distribution $\hat{D}_{\text{mic}}(\omega_h)$ is then convolved with the surface NDF $p_N(n)$. This is similar to normal map filtering and specular shading antialiasing techniques [Olano and Baker 2010; Kaplanyan et al. 2016]. We use the convolution technique described

in Eq. 10 for it and find the best mapping $\boldsymbol{\alpha}_+ = M_3(\boldsymbol{\alpha}, \alpha)$, which is stored as a small 3D table. Because both $\hat{D}_{\text{mic}}$ and $p_N(n)$ are mixtures, the convolution can be carried out per pair of lobes:

$$D_{\text{conv}}(\omega_h) = \int_{\mathbb{S}^2} \hat{D}_{\text{mic}}(\omega_h; n) p_N(n) \, dn = \sum_{i=1}^{k} \sum_{j=1}^{2} w_i m_{ij} D_{\text{conv}}^{ij}(\omega_h),$$

$$\boxed{D_{\text{conv}}^{ij}(\omega_h)} \approx D_{\text{sggx}}(\omega; R^i, \boldsymbol{\alpha}^{ij} = M_3(\boldsymbol{\alpha}^i, M_3^j(\alpha^i))), \qquad (20)$$

where we denote $M_3^j(\alpha^i)$ as a shorthand for fitting a beta distribution for α^i and querying M_3 for the roughness of the j-th D_{mic} lobe (Eq. 19).

5.3 Correction for Conditioned Angular Domain

So far, we have ignored the Heaviside function terms $\mathbb{1}(n \cdot \omega_i)$ and $\mathbb{1}(n \cdot \omega_o)$ in $\mathcal{D}$. Alternatively, when integrating $\mathcal{D} p_{Y_2}(\gamma)$, the angular domain should not be the full sphere $\mathbb{S}^2$, but only a subset conditioned on ω_i and ω_o: $\mathcal{X} = \mathcal{X}_{\omega_i, \omega_o} := \{n \in \mathbb{S}^2 | (n \cdot \omega_i) > 0, (n \cdot \omega_o) > 0\}$. Intuitively speaking, the conditioned domain avoids the incorrect contribution when ω_i and ω_o are from different sides of the surface Otherwise, the ABSDF will suffer from leaking. This complicates the problem because Eq. 20 is not exactly a spherical convolution when the integration domain is $\mathcal{X}$. To keep the efficient convolution-based solution while addressing the potential leaking, we rewrite Eq. 9 and apply the following approximation:

$$\int_{\mathcal{X}} D_{\text{mic}}(\omega_h; n, \alpha) D_{\text{sggx}}(n; R, \boldsymbol{\alpha}) \, dn$$

$$= D_{\text{sggx}}(\omega; R, \boldsymbol{\alpha}_+) \frac{\int_{\mathcal{X}} D_{\text{mic}}(\omega_h; n, \alpha) D_{\text{sggx}}(n; R, \boldsymbol{\alpha}) \, dn}{D_{\text{sggx}}(\omega; R, \boldsymbol{\alpha}_+)} \qquad (21)$$

$$\approx D_{\text{sggx}}(\omega; R, \boldsymbol{\alpha}_+) \boxed{\frac{\int_{\mathcal{X}} D_{\text{mic}}(\omega_h; n, \alpha) \, dn}{\int_{\mathbb{S}^2} D_{\text{mic}}(\omega_h; n, \alpha) \, dn} \left(:= S(\omega_i, \omega_o; \alpha) \right)}.$$

Effectively, we replace the D_{sggx} term in both the right numerator and denominator with a constant term of 1. We name the numerator of S, $\int_{\mathcal{X}} D_{\text{mic}}(\omega_h; n, \alpha) \, dn$, the *shape term*, as it reflects the geometric shape of the angular domain $\mathcal{X}$. Computing the shape term requires integrating a GGX over $\mathcal{X}$, which is a "spherical lune" formed by the intersection of two hemispheres. Note that $\mathcal{X}$ can be decomposed into two spherical triangles, and the problem reduces to integrating a GGX over a spherical triangle, which can be solved in closed form using Linearly Transformed Cosines (LTC) [Heitz et al. 2016]. We precompute the inverse LTC transform into a 1D table $T_{\text{LTC}}^{-1} = M_4(\alpha)$ (different from Heitz et al. [2016], there is only roughness variation in our case). The denominator of S is the normalization term for a GGX in the spherical domain and can be easily precomputed as another 1D table $\int_{\mathbb{S}^2} D_{\text{mic}}(\omega_h; n, \alpha) \, dn = M_5(\alpha)$.

Finally, we utilize the fact that the microfacet shadowing-masking term G is very smooth. Therefore, we simply multiply it to each lobe post convolution. We arrive at the following expression for the yellow highlighted integral in Eq. 16:

$$\boxed{\int_{\Gamma_2} \mathcal{D}p_{Y_2}(\gamma)\,\mathrm{d}\gamma} \approx \sum_{i=1}^{k}\sum_{j=1}^{2} w_i m_{ij} \boxed{D_{\mathrm{conv}}^{ij}(\omega_h)} \boxed{S^{ij}(\omega_i,\omega_o)} G^{ij}(\omega_i,\omega_o),$$

$$S^{ij}(\omega_i,\omega_o) = S(\omega_i,\omega_o;\alpha^{ij}), \qquad (22)$$

$$G^{ij}(\omega_i,\omega_o) = G(\omega_i,\omega_o,R^i,\boldsymbol{\alpha}^{ij}).$$

The expression consists of contributions from all convolved NDF lobes (Eq. 20). Each term is multiplied by its scaling factor to account for the conditioned angular domain (Eq. 21). Eq. 22 can be evaluated in closed form given the surface NDF $p_N(n)$ and the first two moments of roughness α.

5.4 Orientation-varying Parameters

We have previously assumed that β^c, β^m, and β^s are independent of orientation in order to perform the split in Eq. 12 and Eq. 16. To lift this limitation, we notice that both Eq. 13 and Eq. 17 collapse to simple combinations of moments (means and second-order mixed moments) of β^c, β^m, and β^s. Therefore, we extend our formulation by augmenting the moments to be orientation-varying. As an example, when calculating the mean of a parameter $\mathbb{E}[\beta]$ for a particular direction ω, each sample on the surfaces should be weighted to reflect its influence on ω. In other words, each sample is "splatted" to the spherical domain with a spherical function $s(\omega)$ as the kernel, followed by normalization. The *directional moments* can thus be defined as

$$\mathbb{E}_s[\beta]\Big|_\omega = \frac{\int_A \beta_x s(T_x\omega)\,\mathrm{d}x}{\int_A s(T_x\omega)\,\mathrm{d}x}, \quad \mathbb{E}_s[\beta\beta']\Big|_\omega = \frac{\int_A \beta_x \beta_x' s(T_x\omega)\,\mathrm{d}x}{\int_A s(T_x\omega)\,\mathrm{d}x}, \tag{23}$$

where T_x is the local transform at x. The surface NDF is implicitly accounted for by transforming ω into the local coordinate system. The kernel $s(\omega)$ is different for each component of the ABSDF: For the specular component, it is the microfacet distribution; for the diffuse component, it is the SG in Eq. 14. Finally, we query the directional moments at ω_h when evaluating Eq. 13 and Eq. 17.

Now that the moments become orientation-varying, we can no longer store them as simple scalars. In practice, we find that it is usually sufficient to coarsely partition the spherical domain (e.g., 3×3) because the angular frequency usually decreases as the scale of aggregation becomes larger. Each surface sample can then be splatted to the partition during precomputation (see §7.1).

5.5 Summary and Validation

The derivation of our factorized ABSDF is complete at this point. We conclude this section with a brief summary of the complete model with a schematic diagram Fig. 3. The ABSDF (Eq. 7) with the base material (Eq. 11) is the linear sum of a diffuse compoennt (Eq. 12) and a specular component (Eq. 15). The diffuse component is decomposed to a moment term (Eq. 13) and convolution term (Eq. 14). The specular component is decomposed similarly (Eq. 17), but the convolution (Eq. 20) needs to be performed with care given to the base distribution (Eq. 18) and the domain (Eq. 21). Finally, the moments can be augmented to be orientation-varying (Eq. 23). A total of 5 small, scene-independent precomputed tables are utilized in different components as highlighted in Fig. 3.
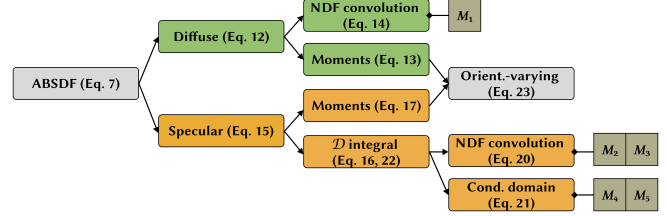

Fig. 3. Schematic diagram of our full factorized ABSDF model. The usages of precomputed tables $M_1, ..., M_5$ are highlighted.

In Fig. 4, we compare our factored ABSDF results to the ground truth and further, the fitting results of the recent neural solution presented by Weier et al. [2023]. Their network (named the "appearance network") consists of a multi-resolution hash grid encoding [Müller et al. 2022] for spatial coordinates, spherical harmonics encodings for incident and outgoing directions, and a multi-layer perceptron to produce the final output. We use the exact same architecture and hyperparameters as Weier et al. [2023] with 8 features per level and 8 degrees of spherical harmonics. We follow a similar training procedure by feeding a large batch of stochastic queries to the network each iteration and optimizing for relative L_2 loss.

The *Helmet* example presents a particularly challenging case with highly glossy anisotropic highlights, which we are able to reconstruct well. On the other hand, the appearance network suffers from various artifacts, including color shift, "blotchiness", mode collapse, and perhaps most significantly, loss of highlights. This could be due to not enough features to capture the spatial variety and that the spherical harmonics encoding cannot handle high frequency signals. The network could potentially benefit from more features and a better directional encoding, but will likely become much larger. For the more diffuse *Palm* example, we are able to capture the dual-mode shape reasonably well thanks to the multi-lobe surface NDF representation. The appearance network performs relatively better on this example but still produces worse accuracy than ours.

In Fig. 5, we demonstrate the necessity of supporting orientation-varying material parameters and the effectiveness of our method. We aggregate a displaced surface with basecolor varying based on orientation and render it from 3 views. The surface exhibits drastically different appearance from different views and our method correctly captures this view-dependent appearance.

6 Correlation-aware Appearance Accumulation

So far, we have presented the definition and an efficient solution for the aggregated appearance of a voxel. In order to render a scene aggregate, we need to accumulate the outgoing radiance contributions of multiple voxels for each pixel. Intuitively speaking, voxel accumulation requires two pieces of information: (1) sub-voxel geometry distribution, and (2) the inter-occlusion across voxels. A core challenge arises from the fact that spatial correlation generally exists in a 3D scene made of surfaces. In the following, we motivate the importance of preserving spatial correlation and discuss our design to model the necessary information for voxel accumulation. With these components, we formulate the process of accumulating voxel contributions into pixel intensity.
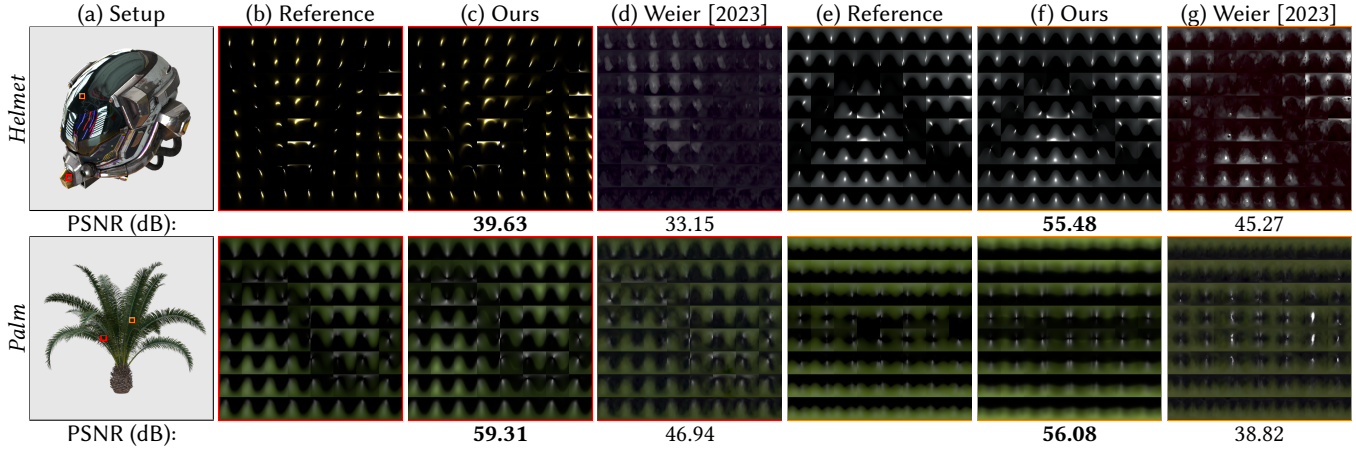
Fig. 4. We select voxels from each scene, highlighted in (a), and compare the ground truth ABSDFs (b)/(e) to our factored ABSDFs (c)/(f) and the appearance network fitted results (d)/(g). Each plot contains 8×8 2D outgoing slices in the lat-long coordinate system with different incident directions. Our results achieve better accuracy both qualitatively and quantitatively with lower RMSE. We encourage readers to zoom in for better comparison. Exposure is adjusted for clarity.
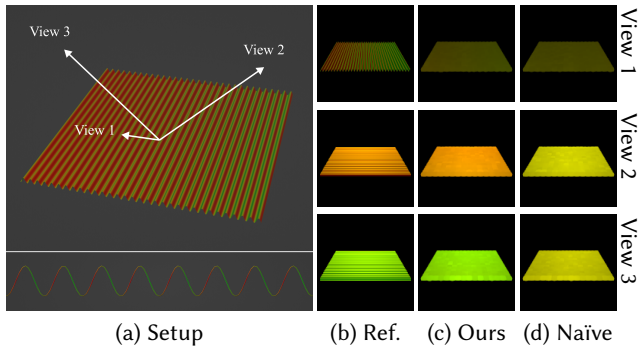

Fig. 5. When a scene has orientation-varying material parameters (a), our method (c) captures the view dependency and matches the reference (b), while ignoring it leads to incorrect results (d).

6.1 Truncated Ellipsoid Primitive

Most volumetric representations produce cube-shaped voxels. For each voxel, it is implicitly assumed that surfaces behave like uncorrelated particles and are independently and uniformly distributed inside. This has been the *de facto* choice and one may argue that with sufficient spatial subdivision, the raw resolution could compensate for the simplicity of this assumption. However, it is important to realize that scene aggregation is more than an image-space signal reconstruction problem. The Nyquist-Shannon sampling rate is thus not twice the image resolution, but twice the highest frequency of geometric details, which can be much higher (if not unbounded) for a scene consisting of hard surfaces. Because it is infeasible to reach such a sampling rate, ignoring the spatial correlation inside each voxel does negatively impact appearance. We demonstrate this by a minimal example in the following.

In this *double-counting* example illustrated in Fig. 6, a simple plane is discretized into diagonally neighboring voxels and is viewed from aside. The voxel size is chosen to be half of the pixel footprint to match the image-space Nyquist-Shannon sampling rate. Because the

voxels have thickness, whenever the film plane is not axis-aligned, some points on the film plane receive contributions from more than one voxel. This is clearly wrong because if we perform ray casting from the film plane, a ray only intersects the ground-truth geometry once. The mismatch is fundamentally because geometry is not distributed uniformly inside a voxel. It could also be interpreted as strong correlation between different voxels: whenever a ray hits one voxel, it should never hit another. However, simple voxels fail to capture this information and result in systematic error. In particular, the error manifests as an objectionable checkerboard-like artifact. We also provide magnified renders with higher image resolution that better illustrate the source of this artifact.

To improve the accuracy of voxel accumulation, we consider ways to support non-uniform intra-voxel distribution. Common approaches that introduce further subdivision within a single voxel, such as using a coverage mask, are essentially no different from brute-force supersampling. They are not cost-effective as we discussed earlier. Instead, we propose to fit a bounding ellipsoid for the geometry in each voxel and define the new voxel primitive as the intersection of the voxel and the ellipsoid. The new *truncated ellipsoid primitive* is much more effective at adapting to different geometry distributions: when the voxel includes a flat surface or a fiber-like thin structure, the primitive now provides a much tighter fit; when the voxel includes unstructured geometry, it falls back to a cube shape. As shown in Fig. 6 (c), the new primitive greatly reduces the artifacts while also producing a tighter object silhouette. Note that we never need to explicitly store the cube-ellipsoid intersection; it is sufficient to store the separate shapes and calculate the properties of the intersection on-the-fly. Because the primitive is a bounding volume, it should support "semi-transparency" to reflect the quantity of the underlying geometry as the geometry is abstracted away (Fig. 8 (a)). We define the *primitive coverage* of a truncated ellipsoid primitive as

$$c(\omega) = \frac{|A|_\omega}{|B|_\omega}, \qquad (24)$$

where $|A|_\omega$ and $|B|_\omega$ are the projected areas of the surfaces bounded by the primitive and the primitive itself, respectively. As will be shown in §6.3, the primitive coverage is useful when accumulating the contributions from multiple voxels, though the numerator $|A|_\omega$ will be canceled and never explicitly needed. For simplicity, we use an efficient Monte Carlo estimator to calculate $|B|_\omega$, the projected area of the intersection of a cube and an ellipsoid, detailed in the supplemental document. Alternatively, it is possible to explicitly calculate the projected area by integrating over the projected contour using Green's theorem. The truncated ellipsoid primitive provides a good trade-off in practice as it is easy to fit and compact to store. See §7.1 for details. Recently, 3D Gaussians have been shown to be effective at representing radiance fields [Kerbl et al. 2023]. Our primitive bears some resemblance to a 3D Gaussian but is ultimately designed for a different purpose. The truncation avoids the ambiguity in defining the inter-occlusion between the otherwise overlapping primitives. It also makes precomputation and rendering more straightforward in practice.
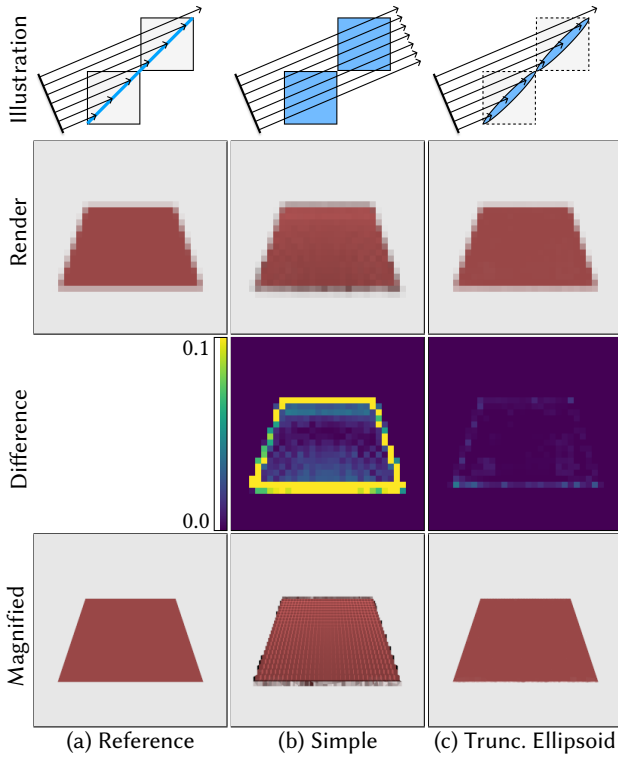

Fig. 6. Compared to the reference, LoD with a simple cube primitive results in a bloated silhouette and worse, structured artifacts on the red plane. With the help of the truncated ellipsoid primitive, our method produces a tighter silhouette and more importantly, artifact free results. We encourage readers to zoom in to better identify the checkerboard-like artifact.

6.2 Aggregated Visibility

In order to accurately accumulate the contributions of voxels, we need to model the visibility between them when the input scene is heterogeneous with a varying degree of spatial correlation. Existing

works show that when spatial correlation exists, transmittance is no longer exponential and cannot be modeled only by extinction coefficients [Jarabo et al. 2018; Bitterli et al. 2018; Vicini et al. 2021]. Hypothetically, it might be appealing to augment the traditional volumetric models with more parameters per voxel. However, simply enhancing the local representation is unlikely to be sufficient because spatial correlation is inherently a long-range effect and it is necessary to model the interaction between voxels. Another attempt is to record and accumulate the coverage masks of voxels [Bako et al. 2023]. This is again similar to brute-force supersampling and requires an impractical amount of memory.

Instead of modeling the visibility by local properties, we propose to model it as a global function. Recall that we produce the split visibility integral in Eq. 6. We further separate the visibility along incident and outgoing directions:

$$\frac{1}{|A|} \int_A V(x, \omega_i)V(x, \omega_o)\, dx \approx \hat{V}(\omega_i)\hat{V}(\omega_o),$$
$$\hat{V}(\omega) = \frac{1}{|A|} \int_A V(x, \omega)\, dx, \tag{25}$$

where $\hat{V}(\omega)$ is the average visibility from points on the surfaces A inside a voxel through the entire scene along ω, and we name it *aggregated interior visibility* (AIV). We illustrate how modeling the global visibility naturally captures spatial correlation by a flatland example shown in Fig. 7. We consider the accumulation of three voxels in two configurations where the results are different due to different types of inter-voxel correlation. In (a), the first two voxels are negatively correlated, while in (b), they are positively correlated. Both configurations have identical per-voxel visibilities (transmittance) $\hat{v}_i$ and coverages c_i as denoted in (c). The traditional volumetric model is thus not able to recognize the correlation and produces an incorrect result by applying the Beer-Lambert law. By explicitly tracking the global AIV $\hat{V}_i$, our method naturally incorporates correlation and produces correct accumulation results.

A slightly different matter arises when a scene aggregate is placed in front of a background or other external objects. In order to correctly blend the contribution from the scene aggregate and the external environment, we need to keep track of another type of aggregated visibility with origins not on surfaces but in the free space. Let P be a pixel footprint in world space observing the scene aggregate from direction ω. Intuitively, we would like to know the average visibility from points on P through the entire scene along ω. However, caution is needed as we should only count the subset of P, P^+, such that rays originated from P^+ actually intersect the scene aggregate primitives, as illustrated in Fig. 8. We define this *aggregated boundary visibility* (ABV) as

$$\hat{V}_b(\omega; P^+) = \frac{1}{|P^+|} \int_{P^+} V(x, \omega)\, dx. \tag{26}$$

As the name implies, the ABV term only needs to be defined on the 2D "boundary" that encloses the 3D scene aggregate. The bound can be any suitable manifold that is reasonably tight such that no external objects intersect it. In our implementation, the ABV term is precomputed and stored on the boundary faces of the voxels. The aggregated visibility functions, AIV and ABV, represent high-dimensional (5D and 4D, respectively) and all-frequency signals.
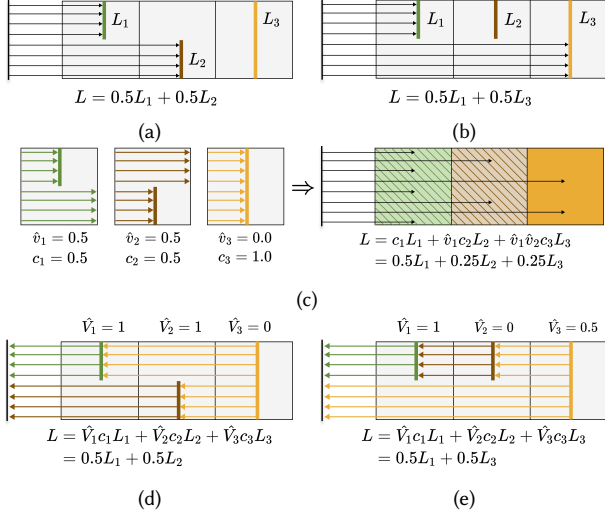
Fig. 7. Even if voxels have identical local properties, different types of inter-voxel correlation leads to different accumulation results ((a) and (b)). While the traditional volumetric model (c) is unable to distinguish these two cases and produces wrong results, our method accounts for this correlation naturally by tracking global visibility ((d) and (e)).

Therefore, we choose to represent them in the Haar wavelet basis. We discuss the details of truncation and further compression strategies in §7.2.

Discussion. In their work, Weier et al. [2023] propose to learn binary visibility by a visibility network and optimize the binary threshold by a weighted F-Measure. The network is trained for *per-voxel* visibility query given a pair of vertices on the boundary of a voxel. During rendering, a ray is partitioned into multiple segments that intersect different voxels and the visibility of each segment is queried separately. We argue that this approach is theoretically limited to handle aggregated inter-voxel correlation due to two reasons. First, there is no aggregation of visibility at all as the network only supports *point-to-point query* without any consideration of filter footprint. Second, this approach essentially assumes no correlation between voxels as both training and inference is performed in a *per-voxel* manner. As discussed and illustrated in Fig. 7, even when individual voxels produce identical statistics, different combinations can still lead to different accumulation visibility. In addition, in their global illumination rendering, an indirect ray is simply spawned from the entry point of current voxel on its boundary with the new scattering direction, which is then used to query the network. This ignores the fact that scattering could happen anywhere inside the voxel and lead to a distribution over exiting positions.

6.3 Evaluating Pixel Intensity

We are now ready to present how to evaluate pixel intensity by accumulating the outgoing radiance of voxels under the far-field assumption. Assuming a pinhole camera and a box pixel reconstruction filter, the intensity of a pixel is the integration of the receiving
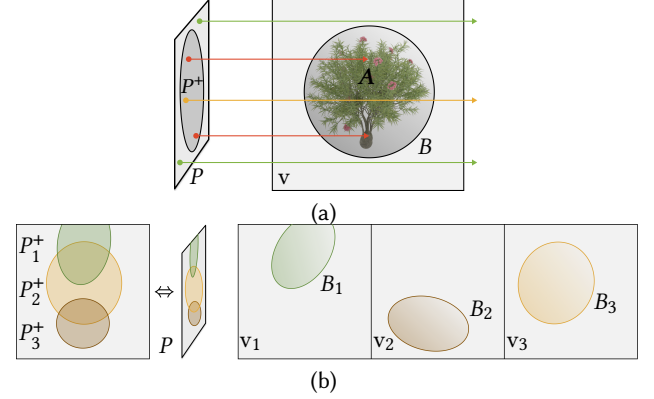

Fig. 8. (a) P^+ is the subset of a pixel footprint P that is covered by a primitive B. A point outside P^+ is guaranteed not to be covered by the underlying geometry (green). For a point inside P^+, it may (red) or may not (yellow) be covered by the underlying geometry. After the geometry is abstracted away, this notion of semi-transparency of B is preserved by the primitive coverage $c(\omega_p)$. (b) When there are multiple voxels, each primitive covers a subset of the pixel footprint P_k^+ and their union becomes P^+: $P^+ = \bigcup_k P_k^+$.

radiance over the footprint P along its direction ω_p:

$$I = \frac{1}{|P|} \int_P L_i(x_p, \omega_p) \, dx_p. \tag{27}$$

We first consider the case of a single voxel and background as illustrated in Fig. 8. The value of the integrand in Eq. 27 depends on whether x_p is "covered" by the truncated ellipsoid primitive B of the voxel, defined as whether the ray spawned from x_p intersects B. Let P^+ be the subset of P that is covered, and $P^- := P \setminus P^+$:

(1) If $x_p \in P^-$, $L_i(x_p, \omega_p)$ simply evaluates to the background radiance $L_b(-\omega_p)$. Note that L_b does not depend on position due to the far-field assumption.

(2) Otherwise, $x_p \in P^+$ and $L_i(x_p, \omega_p)$ is a blend between the outgoing radiance of the voxel, $L_o(-\omega_p)$ as defined in Eq. 3, and the background radiance by the primitive coverage $c(\omega_p)$ as defined in Eq. 24.

Therefore, the pixel intensity becomes

$$I = \frac{1}{|P|} \Big(\int_{P^+} L_o(-\omega_p)c(\omega_p) + L_b(-\omega_p)(1 - c(\omega_p)) \, dx_p + \int_{P^-} L_b(-\omega_p) \, dx_p \Big). \tag{28}$$

Notice that the integrands do not depend on x_p at all because all the quantities are aggregated. We can thus collapse the integrals to

$$I = \frac{|P^+|}{|P|} \Big(L_o(-\omega_p)c(\omega_p) + L_b(-\omega_p)(1 - c(\omega_p)) \Big) + \frac{|P^-|}{|P|} L_b(-\omega_p), \tag{29}$$

where the fraction $|P^+|/|P|$ is the pixel coverage in the common sense (not to be confused with the primitive coverage).

It is straightforward to extend the above formulation to support multiple voxels. Let $\{v_k\}$ be a list of voxels with truncated ellipsoid primitives $\{B_k\}$. The pixel intensity is the sum of the contributions
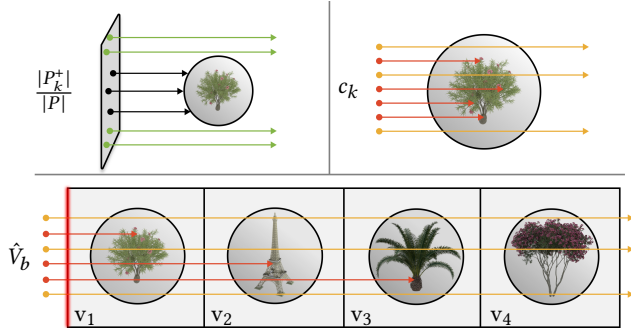
Fig. 9. Illustrating several terms that appear in Eq. 30: The per-voxel pixel coverage (top left), primitive coverage (top right), and the global ABV required to compose the background (bottom).

from all voxels:

$$I = \sum_k \frac{|P_k^+|}{|P|} L_o^k(-\omega_\text{p}) c_k(\omega_\text{p}) \; + \; \frac{|P^+|}{|P|} L_b(-\omega_\text{p}) \hat{V}_b(\omega_\text{p}) +$$
$$\frac{|P^-|}{|P|} L_b(-\omega_\text{p}), \tag{30}$$

where P_k^+ is the subset of P covered by B_k and $P^+ = \bigcup_k P_k^+$. Eq. 30 is similar to Eq. 29 except that the aggregated boundary visibility $\hat{V}_b(\omega_\text{p})$ replaces $(1 - c(\omega_\text{p}))$ in the second term. This is because now we require the average visibility from P^+ through the entire scene to reconstruct the silhouette of the scene aggregate and compose the background. Fig. 9 illustrates several terms in Eq. 30. We note that the voxel accumulation described by Eq. 30 is order-independent. Traditionally, ray marching or back-to-front alpha blending is required for resolving the occlusion (transmittance) between voxels. However, we have already done so in a correlation-aware manner as we have precomputed global visibility as aggregated visibility functions. Therefore, the accumulation during rendering reduces to a simple summation where each voxel modulates its contribution by its AIV. This order-independent property allows efficient parallel implementations.

7 Scene Aggregation Pipeline

In this section, we describe the pipeline of converting an input scene to our aggregated representation and rendering the scene aggregate with suitable LoD selection. In particular, we discuss the practical strategy to compress visibility data, which occupies the majority of the memory footprint.

7.1 Precomputation

The precomputation first performs discretization of the input scene. We utilize the sparsity of typical scenes and discretize the input scene into a hierarchy consisting of multiple levels of sparse grids. Each level doubles the spatial resolution of the one in the previous level. Due to the sparsity, the actual growth rate of non-empty voxel count is only quadratic instead of cubic as a function of resolution. We report the sparsity of all scenes used in Table 2. In our current implementation, each level is precomputed separately. However, it is possible to cache and reuse collected data across levels, which we leave for future optimization.

For each level, the precomputation involves two stages. The first stage precomputes the "interior" of the scene. For each non-empty voxel, we need to acquire the following information:
(1) The total surface area $|A|$ within the voxel (for normalization).
(2) The first two moments of roughness α.
(3) Directional moments of material parameters β^c, β^m, and β^s (Eq. 23) with angular resolution d.
(4) The surface normal distribution $p_N(n)$ function represented by one or a mixture of SGGX distributions with k components (Eq. 8).
(5) The ellipsoid of the truncated ellipsoid primitive as an affine transform.
(6) Wavelet basis coefficients for the aggregated visibility (Eq. 25 and Eq. 26).

Apart from the surface area which can be computed analytically, the rest of the information is estimated via Monte Carlo sampling and ray tracing. We uniformly sample the surfaces within the voxel. The sample budget is a tunable parameter. It should not be too low to avoid noisy estimation. As smaller voxels contain fewer surfaces, we find that one suitable strategy is to allocate sample budget to be inversely proportional to the square of the resolution of current level. This strategy also helps balance the computation cost across different levels. Each surface sample includes position, normal, and material parameters looked up from texture maps. We then proceed to estimate each type of information respectively:

Roughness moments. These are straightforward to compute by moving averages.

Directional moments. An easy way to estimate directional moments is to simply evaluate Eq. 23 at the center direction of each angular grid cell. However, this is prone to aliasing for highly glossy surface samples. Instead, for each surface sample, we warp a low discrepancy sequence to $\mathbb{S}^2$ by the weight kernel g. Then we compute the numerator and the denominator of Eq. 23 for each sequence element and splat them to separate angular grids. We accumulate the contribution for all surface samples and perform the normalization (division) at the end.

Surface normal distribution. For a single SGGX component, we follow the estimation method by Heitz et al. [2015]. For a mixture, we perform a K-means clustering on the surface normal samples and fit each cluster as one component, similar to the process by Zhao et al. [2016]. The initial cluster centers are selected to be away from each other. Furthermore, to avoid undesirable homogeneous clusters, we repeat the fitting for 1 to a maximum of k components and choose the result that yields the highest likelihood. In practice, we find $k \leq 4$ are sufficient for most cases.

Truncated ellipsoid primitive. Finding the optimal minimum volume enclosing ellipsoid is a semidefinite programming problem that can be costly to solve [Todd 2016]. Instead, we compute the approximate minimum bounding ellipsoid for a voxel with a simple heuristic. We first perform Principal Component Analysis (PCA) on the sampled positions and transform them to the unit cube by the eigenvectors. We then compute a bounding sphere and transform it

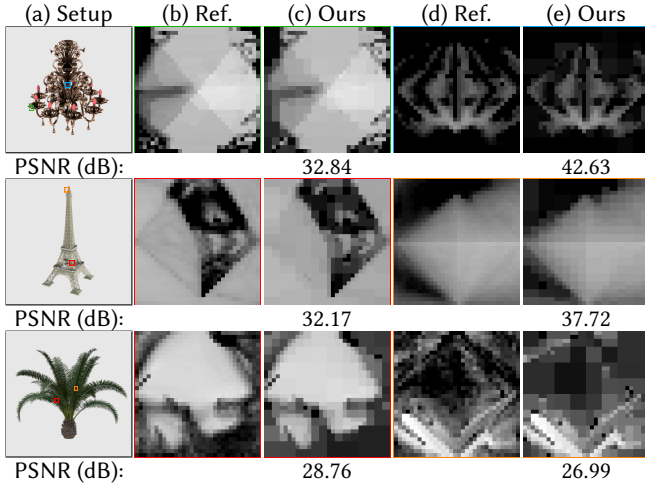| (a) Setup	(b) Ref.	(c) Ours	(d) Ref.	(e) Ours
PSNR (dB):		32.84		42.63
PSNR (dB):		32.17		37.72
PSNR (dB):		28.76		26.99

Fig. 10. Visualizing the AIV terms of selected voxels. Each spherical plot is parameterized by the equal-area mapping [Clarberg 2008]. The wavelet-based projection and compression is able to preserve the high-frequency visibility.

back to world space to obtain a bounding ellipsoid. The resulting ellipsoid is tight enough for our purposes.

Aggregated visibility coefficients. We trace visibility rays with uniformly sampled directions starting from each surface sample and project the results to the Haar wavelet basis. We use the equal-area mapping [Clarberg 2008] to parameterize the spherical domain. Again, the visibility sample rate should not be too low to avoid noisy estimation. However, when there are a sufficient number of surface samples, the cost of tracing visibility can be amortized. In practice, we find 16-64 rays per surface sample is enough. In Fig. 10, we visualize the AIV terms and compare our compressed terms to the references.

The second stage of the precomputation handles the ABV term. Recall that this term is only defined on the boundary of the entire scene. Therefore, after scene discretization, we precompute it for the boundary faces of the voxels. The list of boundary faces can be determined by one simple flood fill iteration. For each boundary face, we consider all directions in its inward facing hemisphere. For each direction, we cast visibility rays with origin uniformly sampled on the face to estimate the average visibility. We use concentric mapping [Shirley and Chiu 1997] to parameterize the hemispherical domain and obtain a 2D visibility map. The map is then projected to the Haar wavelet basis. The ABV contains a high-frequency signal as it is responsible for reconstructing the silhouette of the scene. In practice, we choose a relatively high angular resolution of 64^2 for accurate reconstruction. Fig. 11 shows the accurate coverage reconstruction with our ABV term.

7.2 Compression Strategy for Visibility Data

In order to preserve all-frequency information, we represent both types of visibility: the aggregated interior visibility (AIV), and the aggregated boundary visibility (ABV) by wavelet coefficients. Typically, we are able to perform nonlinear approximation and truncate
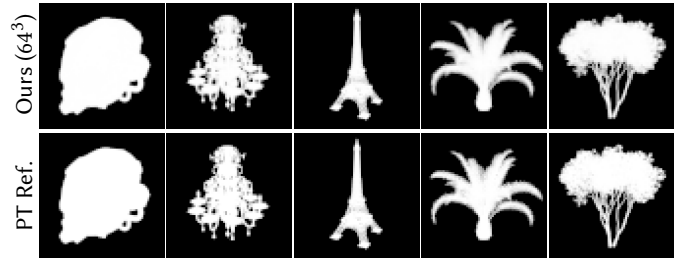

Fig. 11. Visualizing coverage for all five scenes in Fig. 13. Our ABV term accurately reconstructs partial coverage (transmittance) for different types of scenes after compression.

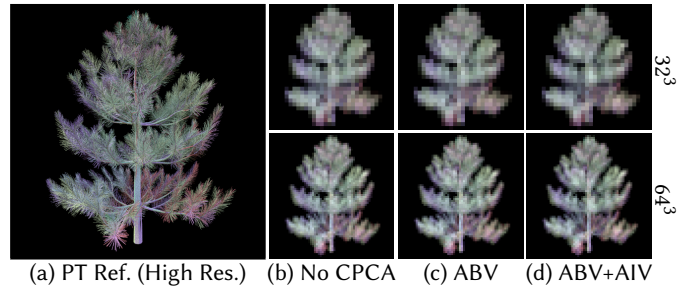(a) PT Ref. (High Res.) (b) No CPCA (c) ABV (d) ABV+AIV

Fig. 12. We apply CPCA to compress both the aggregated boundary (ABV) and interior visibility (AIV) data. In this example, we reach a $\sim 4\times$ compression ratio while having little impact on the visual quality.

a large number of coefficients while preserving good quality. However, this is not enough when the angular resolution is high. We typically use 32^2 resolution for AIV and 64^2 resolution for ABV as it is responsible for reconstructing a sharp silhouette. Even with a typical 90% to 95% truncation rate, the memory cost can still be high as the spatial resolution grows. Therefore, we further apply Clustered Principal Component Analysis (CPCA), which is proven to be effective at compressing basis coefficients [Sloan et al. 2003; Liu et al. 2004]. One performance issue for CPCA is that PCA has cubic time complexity and quadratic space complexity with respect to input data matrix size. Thus, it becomes impractical to directly apply it to a fine LoD level. We apply a simple heuristic by dividing a level into individual blocks of no more than 64^3 and applying CPCA to each block separately. This works well in practice, since a large extent of spatial locality is still preserved in each block that can be exploited by CPCA. It is possible to develop more sophisticated methods to scale CPCA or compress coefficients which is left for future work.

In Fig. 12, we validate the effectiveness of our current CPCA-based compression. For this *Colortree* scene, we compress AIV to 30 clusters each with 10 representatives and ABV to 30 clusters each with 60 representatives. Each representative still goes through coefficient truncation after CPCA. Overall, we gain an extra $\sim 4\times$ compression ratio without negatively impacting the visual quality.

7.3 Rendering with LoD Selection

The rendering of our scene aggregate follows Eq. 30. For each pixel, we need to determine the list of voxels $\{v_k\}$ whose primitives $\{B_k\}$ cover the pixel footprint P and compute the pixel coverage $|P_k^+|/|P|$

Algorithm 1 Rendering a scene aggregate S given a cone with center ray r and aperture θ.

1: **function** RENDER(S, r, θ)
2: $\quad t, t_{\max}$ = intersect(S.bound, r)
3: $\quad s$ = base voxel size of S
4: $\quad \omega$ = direction of r
5: $\quad I = 0, \hat{V}_b = 1$
6: $\quad$ **while** $t < t_{\max}$ **do**
7: $\quad\quad$ level = floor(log2(tan(0.5 $\cdot \theta$)$\cdot t/s$))
8: $\quad\quad \mathrm{v}_k, \triangle t$ = DDAToNextVoxel(r, t, S[level].grid)
9: $\quad\quad$ **if** v_k is on the boundary of S **then**
10: $\quad\quad\quad \hat{V}_b$ = EvaluateABV(t, ω)
11: $\quad\quad$ **end if**
12: $\quad\quad$ **if** intersect(B_k, r) **then**
13: $\quad\quad\quad L_o = 0$
14: $\quad\quad\quad$ **for** $i = 1$ to m **do**
15: $\quad\quad\quad\quad L_o$ += $(1/m) \cdot$ MISDirectLighting(v_k)
16: $\quad\quad\quad\quad\quad\quad \triangleright$ incident AIV by $\hat{V}_i$ = EvaluateAIV(v_k, ω_i)
17: $\quad\quad\quad$ **end for**
18: $\quad\quad\quad \hat{V}_o$ = EvaluateAIV($\mathrm{v}_k, -\omega$)
19: $\quad\quad\quad I$ += $L_o \cdot \hat{V}_o \cdot |A_k| / |B_k|_\omega$
20: $\quad\quad\quad\quad\quad\quad\quad\quad\quad \triangleright$ See supplemental document
21: $\quad\quad$ **end if**
22: $\quad\quad t$ += $\triangle t$
23: $\quad$ **end while**
24: $\quad \hat{L}_b$ = EvaluateBackground($-\omega$)
25: $\quad I$ += $\hat{L}_b \cdot \hat{V}_b$
26: $\quad$ **return** I
27: **end function**

---

for each k. In our current implementation, we choose to compute it by multi-sampled ray casting. Each ray traverses the discretized scene by a digital differential analyzer (DDA). For each encountered voxel, we compute and accumulate its outgoing radiance L_o^k with the primitive coverage c_k (Eq. 24). Note that the traversal can be in arbitrary order, which enables possible rasterization-based approaches.

To utilize the different LoDs included in the hierarchy, we can enhance the above procedure by associating each ray with a cone aperture that covers the pixel footprint, akin to ray differentials [Igehy 1999]. During the traversal of each ray, we determine the LoD level by the cross section size of the cone. In our current implementation, we switch to a coarser LoD only at the boundary of the coarser voxels for efficiency. A continuous LoD blending scheme is possible but more costly. Algo. 1 provides pseudocode for the rendering procedure.

To enable next event estimation (NEE) with multiple importance sampling (Algo. 1, line 15), we develop a straightforward importance sampling routine for our factored ABSDF $\hat{f}_{\text{novis}}$ (Eq. 7) as follows:

(1) Pick one component between the specular and diffuse components. This can be done simply by uniform sampling.

(2) For the specular component, we first pick one convolved lobe from **??** based on the lobe weights w_j. Then we sample the corresponding SGGX distribution.

(3) For the diffuse component, note that we cannot directly sample a convolved lobe from Eq. 14 because κ_{ij} cannot be determined without ω_i. Therefore, we resort to a simple strategy by assuming a fixed κ during importance sampling. The rest is similar to the specular case: we pick one convolved lobe (but with the fixed κ) and sample the corresponding SGGX distribution.

The corresponding PDF computation is also straightforward. The sample budget for NEE is decoupled from the ray casting sample budget.

8 Results and Discussion

In this section, we provide rendering results produced by our scene aggregation pipeline and detailed comparison to existing techniques. We implement our method in a custom CPU renderer using Embree [Wald et al. 2014] as the ray tracing backend for precomputation and reference generation. The sparse hierarchical data structure is implemented using OpenVDB/NanoVDB [Museth 2013, 2021]. All timings are measured on a desktop machine with an Intel i9-13900K CPU and 64 GB of main memory. Unless otherwise stated, we use path-traced images with direct illumination as reference.

We compare our method to three current state-of-the-art methods: the hybrid mesh-volume LoD method (HybridLoD) [Loubet and Neyret 2017], the non-exponential transmittance volumetric model (NonExp) [Vicini et al. 2021], and the deep appearance prefiltering (DAP) [Bako et al. 2023]. For HybridLoD, we use the official implementation provided by authors with modifications for asset loading purposes. For NonExp, we re-implemented the method based on the paper as the source code is not available. For DAP, we used the authors' pre-trained results as training is prohibitively expensive and requires a GPU cluster. We provide different images from references and provide root mean squared error (RMSE) to evaluate the quality of each method. In addition, we provide a supplementary video with varying magnification levels, camera rotation, and lighting conditions to demonstrate the temporal stability of our method.

Rendering Quality Comparison. In Fig. 13, we compare our method to HybridLoD and NonExp on a set of scenes with varying geometric and material characteristics. For each scene, we show the rendered results using 2 different LoD scales, 32^3 and 64^4. The image resolutions are 32^2 and 64^2, chosen such that a voxel roughly projects to the footprint of a pixel. High resolution references are provided to better visualize the setup. The *Helmet* scene has relatively low geometric complexity but it consists of large specular surfaces which are traditionally challenging for LoD methods. The *Chandelier* scene has intricate geometric structure with varying degree of curvature that produces anisotropic highlights. The *Tower* scene features organized thin structures that lead to correlated partial occlusion. Finally, the *Palm* and the *Oleander* scene have larger complexity with both unstructured (leaves) and structured (trunk) geometry.

For all scenes, our results achieve superior quality and produce closer matches to references, as can be verified by the difference images and the RMSE errors. HybridLoD tends to produce bloated, over transparent results, which is especially noticeable at coarser LoD resolution. This could be due to both misclassification (too much volume) and the neglect of correlation. Moreover, the mesh simplification process could undesirably alter the curvature of the

Table 2. Scene Configuration. For all scenes: Surface NDF mixture component count is set to $k \leq 4$; AIV is recorded at 32^2 angular resolution; ABV is recorded at 64^2 angular resolution. At most top 10% basis coefficients are kept. More are truncated as long as the reconstruction preserves more than 95% accuracy. *Instanced triangle count. †This scene is composed of instances of multiple aggregated objects. We only report the largest object.

Scene	Original		Ours			
	#Tris	Memory	Max Res.	Occupancy	Total Mem.	Precomp. Time
Helmet	15K	7.7 MB	256^3	2.36%	155.1 MB	1583 sec
Chandelier	106K	11.7 MB	256^3	1.39%	180.6 MB	748 sec
Tower	453K	45.7 MB	256^3	1.66%	255.8 MB	1108 sec
Palm	2.2M	349.6 MB	256^3	1.25%	216.2 MB	1345 sec
Oleander	2.7M	398.8 MB	256^3	3.71%	327.6 MB	3537 sec
Coral Reef	4.1M	513.0 MB	256^3	3.18%	209.5 MB	1624 sec
Forest	16.2M (*175.2M)	2.5 GB	512^3	2.86%	1.06 GB	16064 sec
Metropolis	88.6M (*301.8M)	46.9 GB	$†256^3$	$†7.01\%$	5.33 GB	18254 sec

original geometry, causing loss of highlights (*Helmet* and *Chandelier*). NonExp achieves better quality than HybridLoD in general, but still suffers from several issues. The transmittance optimization accounts for some correlation but is usually not perfect, as shown in the *Chandelier* renders (too leaky) and the *Palm* renders (too opaque). The method ignores the complexity in material and results in glossy appearance mismatch (*Helmet* and *Chandelier*). Ultimately, the empirical exponential-linear blending model is unlikely to satisfy all constraints required to match transmittance for all directions. In addition, we find that it is highly sensitive to the empirical ray offset parameter as a different value drastically alters brightness. We follow the authors' suggestion and offset scattered rays by one voxel for all results.

Comparison to DAP. In Fig. 14, we provide a separate comparison to DAP as we only use their pretrained asset. The *Oak* scene presents two difficulties including the glinty appearance from the highly glossy material and the hard shadow cast by a directional light. Our method is able to capture the highlight accurately, but fails to reconstruct the hard shadow perfectly due to the coefficient truncation and compression error. This can be alleviated with more conservative truncation/compression parameters at the cost of a larger memory footprint. Overall, our method is able to reach a comparable visual quality (slightly better in terms of RMSE). We emphasize that our method only requires a fraction of precomputation time, memory cost, and rendering time to reach such quality.

Complex Scenes. We showcase the practicality of our method by demonstrating results on significantly more complex scenes. Each scene shown in this part features a collection of assets with multiple geometric parts and materials. We compare our results to references but not to other methods, because they require either non-trivial engineering effort or excessive precomputation budget to support assets at this scale.

The *Coral Reef* scene in Fig. 1 includes a variety of geometry (flat surfaces and unstructured details) and materials (glossy and diffuse). The environment light features a dynamic range of 80,000:1 and produces strong highlights on glossy surfaces. Despite the challenging configuration, our results accurately preserve the appearance across different scales. The insets show how our method deals with a particularly challenging part with numerous thin glossy branches.

Table 3. Statistics of the rendering results in Fig. 13. Memory consumption and render times are measured using 64^3 LoD resolution and 1024 samples per pixel for all methods.

Helmet	PT Ref.	Ours	HybridLoD	NonExp
Mem. (MB)	7.7	9.7	5.1	1.0
Time (sec)	1.37	1.68	23.78	57.36
RMSE	—	0.158	0.302	0.238
Chandelier	PT Ref.	Ours	HybridLoD	NonExp
Mem. (MB)	11.7	6.4	6.6	0.5
Time (sec)	1.46	1.74	32.33	51.08
RMSE	—	0.075	0.112	0.099
Tower	PT Ref.	Ours	HybridLoD	NonExp
Mem. (MB)	45.7	9.6	4.3	0.7
Time (sec)	1.66	1.20	19.72	37.13
RMSE	—	0.022	0.050	0.038
Palm	PT Ref.	Ours	HybridLoD	NonExp
Mem. (MB)	349.6	7.3	4.0	0.5
Time (sec)	2.93	1.81	30.50	56.79
RMSE	—	0.024	0.109	0.098
Oleander	PT Ref.	Ours	HybridLoD	NonExp
Mem. (MB)	398.8	12.8	4.1	1.4
Time (sec)	4.56	5.12	84.78	192.39
RMSE	—	0.027	0.090	0.045

At the coarsest scale, even our multi-lobe NDF does not have sufficient angular resolution to resolve all the highlights, resulting in a slightly darker look. However, this is alleviated at finer scales.

The *Forest* scene in Fig. 16 features the largest single-object geometric complexity. It has 16.2 million unique triangles and 175.2 million after instancing. Our results remain close to references at the three LoD scales shown (128^3, 256^3, and 512^3). We note that the memory cost of our method is agnostic to whether or not the original scene contains instanced geometry.

Finally, our method supports assembly of multiple aggregated objects and instancing for even larger scenes. The *Metropolis* cityscape in Fig. 1 is composed of 82 unique aggregated objects and 270 instances. Each instance selects its level based on its screen-space projection size so that content far away can be rendered with coarser LoDs accordingly (Fig. 1, right). Different instances are treated as uncorrelated and ABVs are used to compute the partial occlusion
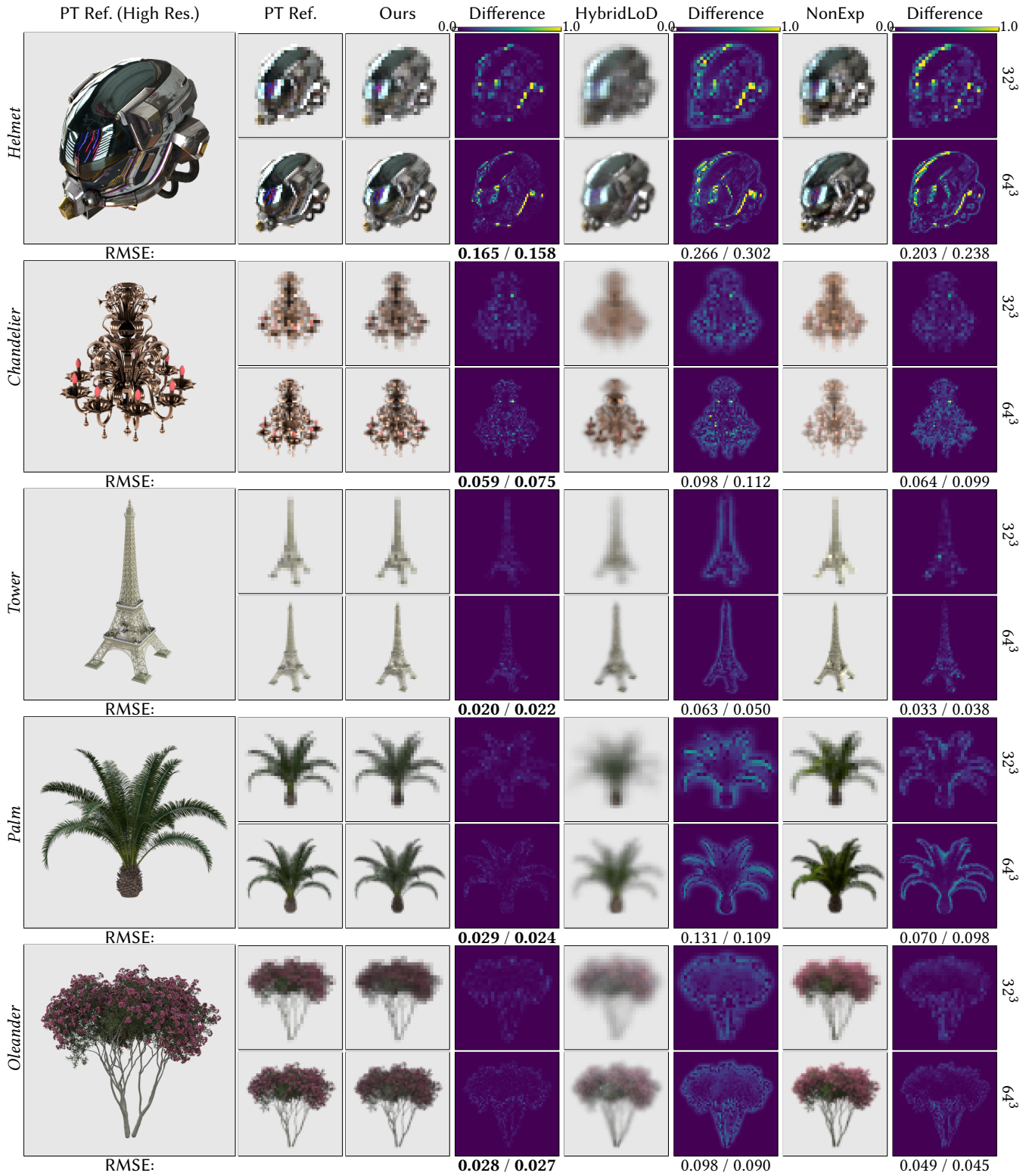
Fig. 13.   Rendering results by our and existing methods on a variety of scenes. All images are rendered using 1024 samples per pixel. Each result is compared to the corresponding reference and the difference image is displayed on the side with RMSE provided. Our results achieve superior quality for all five scenes.
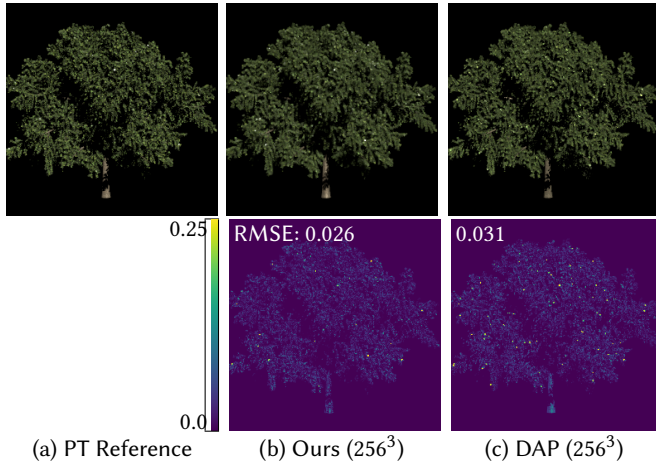
(a) PT Reference	(b) Ours (256^3)	(c) DAP (256^3)

Fig. 14. We compare our method to DAP with their pretrained *Oak* scene. Despite the small imperfection of the hard shadow cast on the tree trunk due to visibility coefficient truncation and compression, our result overall reaches similar visual quality.

between them. This is where the LoD approach truly shines as our representation provides nearly an order-of-magnitude of memory saving. In fact, we are not even able to generate reference for this scene in our testing environment: The original scene costs 46.9 GB alone and when taking into account the auxiliary data structures such as mipmaps and the BVH, the total memory would have exceeded the 64 GB main memory of our testing machine, requiring out-of-core rendering. Our representation only costs 5.33 GB and Fig. 1 takes 1300 seconds to render at 1280×540 resolution with 2048 samples per pixel.

Performance. In Table 2, we report the configuration for all scenes, including both the original version and our representation. It is clear that all scenes have a large degree of sparsity. As expected, the memory footprint of our representation is largely independent of the original scene complexity. While our method cannot beat the explicit representation for small scenes, the asymptotical benefit is evident for larger scenes. Recall that the memory footprint scales quadratically with respect to LoD resolution. If a model only occupies a small portion of the rendered image, then only a coarser LoD level with much smaller memory is needed. It is possible to further reduce the memory footprint with a more optimized implementation, e.g., by quantizing the stored data. GPU ray tracing is likely to reduce the precomputation time by an order of magnitude.

In Table 3, we compare the memory consumption and rendering times required by our method, HybridLoD, and NonExp. At a relatively modest resolution of 64^3, all methods reduce the memory consumption, especially for more complicated models. However, our method does require more memory than HybridLoD and NonExp mainly due to the high-dimensional aggregated visibility data even after compression. Performance wise, none of the methods show significant advantage over the PT baseline at equal samples. In fact, HybridLoD and NonExp are significantly slower, likely due to the complexity of (nested) ray marching or sampling. We note that the baseline is backed by highly optimized ray tracing kernels from Embree, and our CPU re-implementation of NonExp is relatively

unoptimized. Our method is much more comparable to the baseline, as it is simpler with precomputed aggregated visibility (Eq. 30). The equal-sample comparison could be unfavorable to our and other LoD methods because they are already prefiltered and should require fewer samples to reach the same or similar quality. In the following, we assess how our method improves efficiency under equal time.

Thanks to appearance aggregation, we are able to render complex scenes efficiently. Fig. 17 shows equal-time rendering comparison for the *Forest* scene. The performance improvement comes from two aspects: First, we avoid spending a large number of samples to trace explicit geometry as our representation is already anti-aliased. This allows us to perform splitting and allocate more samples for lighting (Algo. 1, line 14). Moreover, we do not need to trace shadow rays either, since the visibility information is readily available from the precomputed AIV. The variance reduction is modest, as ray tracing is highly optimized by Embree. Again, our rendering speed can be further improved with a GPU implementation.

Ablation Study. In Table 4 and Fig. 18, we conduct an ablation study on the impact of parameters to accuracy and cost. For each model, we precompute it with the "maximum" configuration to serve as the control, where up to 4 surface NDF lobes are allowed and no compression is applied at all. We then vary each parameter and assess its affect on rendering quality, memory requirement, and time compared to the control. The benefit of using multiple NDF lobes is more prominent in the *Chandelier* scene to capture the glossy base material together with curved surfaces; it is not obvious in the more diffuse *Oleander* scene. For both types of aggregated visibility, 10% coefficient truncation only introduces barely recognizable error, while 1% truncation results in visible inaccuracy. Finally, by keeping only 1 CPCA representative, CPCA is reduced to simple vector quantization and approximates per-cluster subspaces poorly. The visibility data is high dimensional and thus compression parameters greatly affects the final memory requirement. All parameters affect the shading cost but it is minor compared to voxel traversal and intersection test.

Limitations. Our method has several limitations that could serve as fruitful topics for future research. As prefaced, so far we have been focusing on direct illumination. In order for a scene aggregate to support global illumination, multiple scattering between different parts of the scene should be modeled. This brings new challenges as discretizing and aggregating individual regions will inevitably lose the information about how different regions interact with each other. One possible approach is to precompute and aggregate the entire transport from the external environment to a given region. The definition of ABSDF should be extended accordingly in this case.

Two more limitations stem from the separate approximations made in Eq. 6 and Eq. 25. Eq. 6 assumes independence between visibility and material in a single voxel which could lead to certain artifacts as illustrated in Fig. 19. However, this is alleviated as the spatial resolution grows, since the correlated parts are more likely to be grouped into different voxels. Eq. 25 assumes independence between visibility along two directions. This could lead to incorrect occlusion when, for example, the camera and the light source are collocated. To the best of our knowledge, compactly representing the
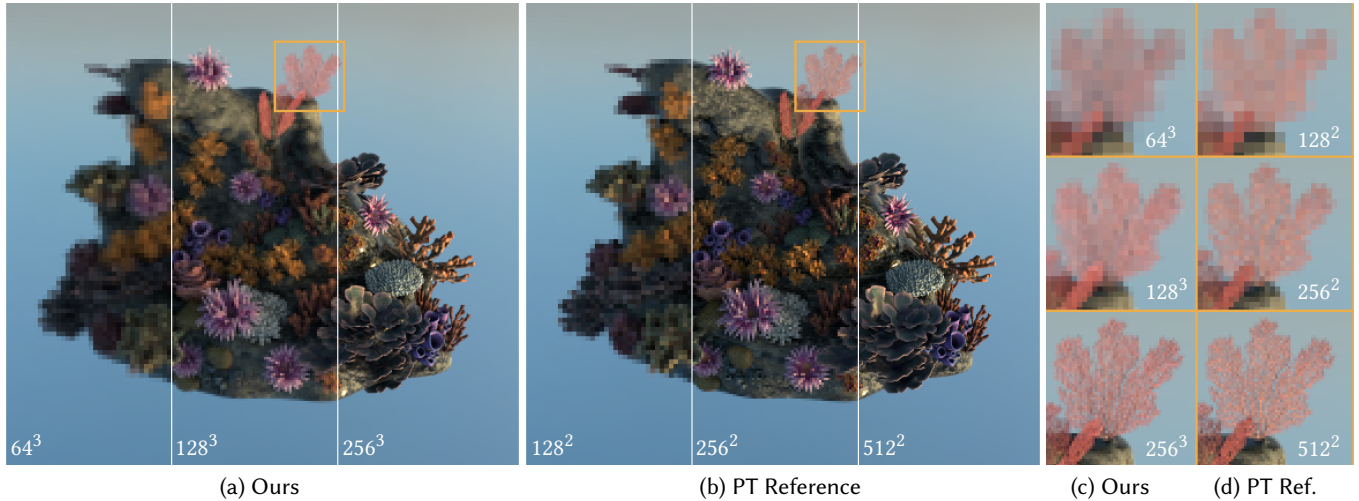
Fig. 15. Our method captures the complex visual appearance of the *Coral Reef* scene that consists of a variety of geometry and materials. We show (a) the renders with our representation at 3 scales (64^3, 128^3, and 256^3) and compare them to (b) path-traced (PT) references. Image resolutions are chosen such that one voxel approximately projects to the footprint of a single pixel (128^2, 256^2, and 512^2). We highlight a challenging part that features partial transparency and a glossy material on the right.

Table 4. Ablation study analyzing the impact of parameters on appearance and cost. Given the "maximum" configuration with up to 4 surface NDF lobes and **no** compression, we change each paramter and show how it affects accuracy, memory requirement, and render time. Please refer to Fig. 18 for rendered results.

Scene		Maximum	1 NDF Lobe	10% AIV	1% AIV	10% ABV	1% ABV	1 CPCA Rep.
Chandelier 64^3	Mem. (MB)	70.1	69.5 (0.99×)	48.9 (0.70×)	46.8 (0.67×)	31.2 (0.44×)	27.3 (0.39×)	27.2 (0.37×)
	Time (sec)	1.91	1.87 (0.98×)	1.91 (1.00×)	1.62 (0.85×)	1.81 (0.95×)	1.78 (0.93×)	1.85 (0.97×)
	RMSE	0.070	0.077	0.070	0.081	0.071	0.101	0.078
Chandelier (alt. view)	Time (sec)	2.86	2.72 (0.95×)	2.83 (0.99×)	2.40 (0.84×)	2.77 (0.97×)	2.72 (0.95×)	2.63 (0.92×)
	RMSE	0.242	0.322	0.247	0.290	0.246	0.266	0.251
Oleander 64^3	Mem. (MB)	125.8	124.5 (0.99×)	66.6 (0.53×)	60.7 (0.48×)	154.7 (0.62×)	77.4 (0.58×)	71.2 (0.57×)
	Time (sec)	5.13	5.03 (0.98×)	5.13 (1.00×)	4.57 (0.89×)	5.08 (0.99×)	5.18 (1.01×)	5.14 (1.00×)
	RMSE	0.025	0.025	0.026	0.028	0.032	0.116	0.053
Oleander (alt. view)	Time (sec)	8.06	7.90 (0.98×)	7.93 (0.98×)	6.77 (0.84×)	7.66 (0.95×)	7.90 (0.98×)	7.81 (0.97×)
	RMSE	0.046	0.046	0.047	0.047	0.055	0.088	0.070

general 4D correlated bidirectional visibility remains an open problem for LoD techniques. In practice, we find that this assumption rarely causes noticeable artifacts.

Finally, it would be desirable to further support material models beyond the Disney BRDF. For example, foliage often exhibits non-negligible subsurface scattering effects. One simple extension to our current ABSDF factorization could be to model an extra diffuse transmission component.

9 Conclusion

We present an efficient scene appearance aggregation method for LoD rendering. Our method is based on a novel formulation for far-field scene aggregation with the definition of ABSDF, which captures the aggregated appearance of all surfaces within a volume. We develop a closed-form factorization of the ABSDF that supports all-frequency and view-dependent effects with handy evaluation and sampling procedures. Our representation naturally accounts for long-range correlation by recording two types of global visibility,

the aggregated interior visibility and the aggregated boundary visibility. Our truncated ellipsoid primitive improves the preservation of local correlation compared to the naïve cubic primitive. We have demonstrated the accuracy of our method on a variety of scenes with different geometric and material characteristics and its scalability to large, complex scenes. Our results achieve higher quality than those from state-of-the-art LoD techniques. While our implementation is far from optimized, we can already show the asymptotic advantages of our representation in terms of memory footprint and rendering speed compared to the original representation. We believe our work is highly relevant to improving the scalability of physically based rendering, enabling the generation of richer, more realistic 3D content.

References

Michael Ashikhmin, Simon Premoze, and Peter Shirley. 2000. A microfacet-based BRDF generator, Judith R. Brown and Kurt Akeley (Eds.). *ACM Trans. Graph.*, 65–74. https://doi.org/10.1145/344779.344814

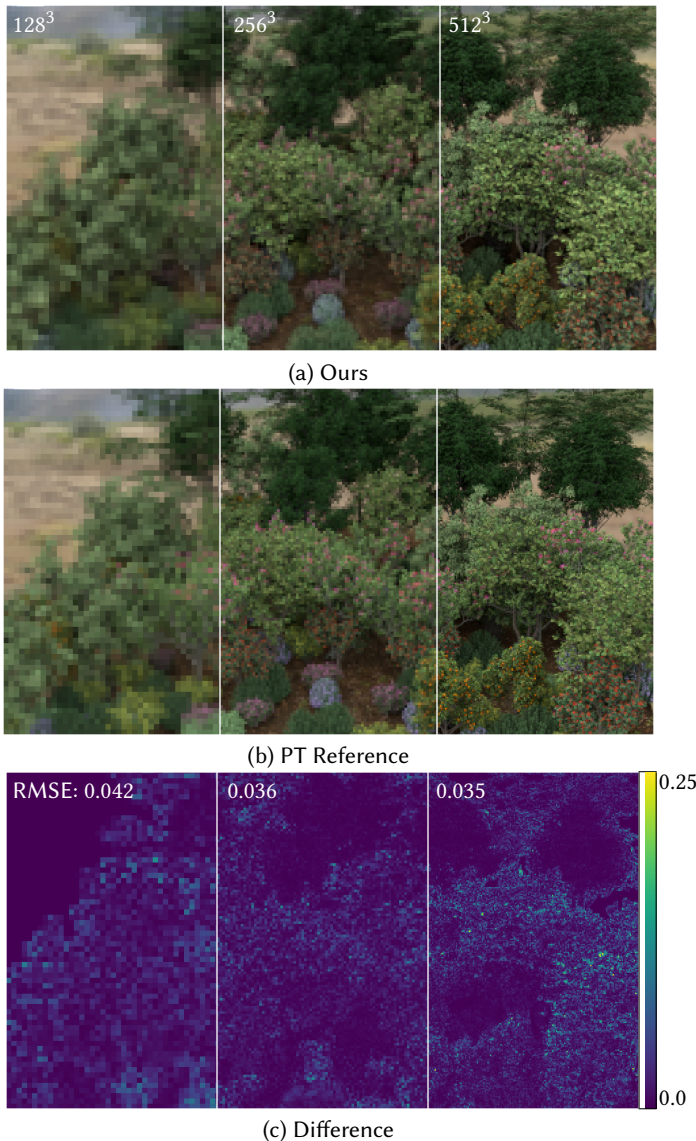(a) Ours

(b) PT Reference

(c) Difference

Fig. 16. Our results faithfully preserve the partial coverage, visibility, and appearance of the large *Forest* scene at different scales. The difference images are scaled for clarity.

Fig. 17. Equal-time rendering comparison. Both images are rendered for 1 second. Our method performs splitting to improve efficiency by allocating 5 NEE samples per ray sample.

1145/3272127.3275103

Brent Burley. 2012. Physically-based shading at disney. *ACM Trans. Graph.* 2012, 1–7.

Subrahmanyan Chandrasekhar. 1960. *Radiative Transfer.* New York.

Petrik Clarberg. 2008. Fast Equal-Area Mapping of the (Hemi)Sphere using SIMD. *J. Graph. Tools* 13, 3 (2008), 53–68. https://doi.org/10.1080/2151237X.2008.10129263

Michael F Cohen and John R Wallace. 1993. *Radiosity and realistic image synthesis.* Morgan Kaufmann.

Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. 2007. Stochastic simplification of aggregate detail. *ACM Trans. Graph.* 26, 3 (2007), 79. https://doi.org/10.1145/1276377.1276476

Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann. 2009. GigaVoxels: ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics, SI3D 2009, February 27 - March 1, 2009, Boston, Massachusetts, USA,* Eric Haines, Morgan McGuire, Daniel G. Aliaga, Manuel M. Oliveira, and Stephen N. Spencer (Eds.). ACM, 15–22. https://doi.org/10.1145/1507149.1507152

Jonathan Dupuy, Eric Heitz, and Eugene d'Eon. 2016. Additional Progress Towards the Unification of Microfacet and Microflake Theories. In *27th Eurographics Symposium on Rendering, Rendering - Experimental Ideas & Implementations, EGSR 2016, EI&I Track, Dublin, Ireland, June 22-24, 2016,* Elmar Eisemann and Eugene Fiume (Eds.). Eurographics Association, 55–63. https://doi.org/10.2312/sre.20161210

Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. 2013. Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph.* 32, 6 (2013), 211:1–211:11. https://doi.org/10.1145/2508363.2508422

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics, G. Scott Owen, Turner Whitted, and Barbara Mones-Hattal (Eds.). *ACM Trans. Graph.,* 209–216. https://doi.org/10.1145/258734.258849

Alban Gauthier, Robin Faury, Jérémy Levallois, Théo Thonat, Jean-Marc Thiery, and Tamy Boubekeur. 2022. Mipnet: Neural normal-to-anisotropic-roughness mip mapping. *ACM Transactions on Graphics* 41, 6 (2022), 1–12.

Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. 2007. Frequency domain normal map filtering. *ACM Trans. Graph.* 26, 3 (2007), 28. https://doi.org/10.1145/1276377.1276412

Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification.. In *EGSR (DL).* 85–97.

Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX microflake distribution. *ACM Trans. Graph.* 34, 4 (2015), 48:1–48:11. https://doi.org/10.1145/2766988

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–8.

Eric Heitz and Fabrice Neyret. 2012. Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. In *Proceedings of the EUROGRAPHICS Conference on High Performance Graphics 2012, Paris, France, June 25-27, 2012,* Carsten Dachsbacher, Jacob Munkberg, and Jacopo Pantaleoni (Eds.). Eurographics Association, 125–134. https://doi.org/10.2312/EGGH/HPG12/125-134

Hugues Hoppe. 1996. Progressive Meshes, John Fujii (Ed.). *ACM Trans. Graph.,* 99–108.

Homan Igehy. 1999. Tracing Ray Differentials, Warren N. Waggenspack (Ed.). *ACM Trans. Graph.,* 179–186. https://doi.org/10.1145/311535.311555

Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. 2012. Interactive bi-scale editing of highly glossy materials. *ACM Trans. Graph.* 31, 6 (2012), 144:1–144:7. https://doi.org/10.1145/2366145.2366163

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2022. NeRF-Tex: Neural Reflectance Field Textures. *Comput. Graph. Forum* 41, 6 (2022), 287–301. https://doi.org/10.1111/cgf.14449

Steve Bako, Pradeep Sen, and Anton Kaplanyan. 2023. Deep Appearance Prefiltering. *ACM Trans. Graph.* 42, 2 (2023), 23:1–23:23. https://doi.org/10.1145/3570327

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021.* IEEE, 5835–5844. https://doi.org/10.1109/ICCV48922.2021.00580

Sai Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milos Hasan, Yannick Hold-Geoffroy, David J. Kriegman, and Ravi Ramamoorthi. 2020. Neural Reflectance Fields for Appearance Acquisition. *CoRR* abs/2008.03824 (2020). arXiv:2008.03824

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. A radiative transfer framework for non-exponential media. *ACM Trans. Graph.* 37, 6 (2018), 225. https://doi.org/10.
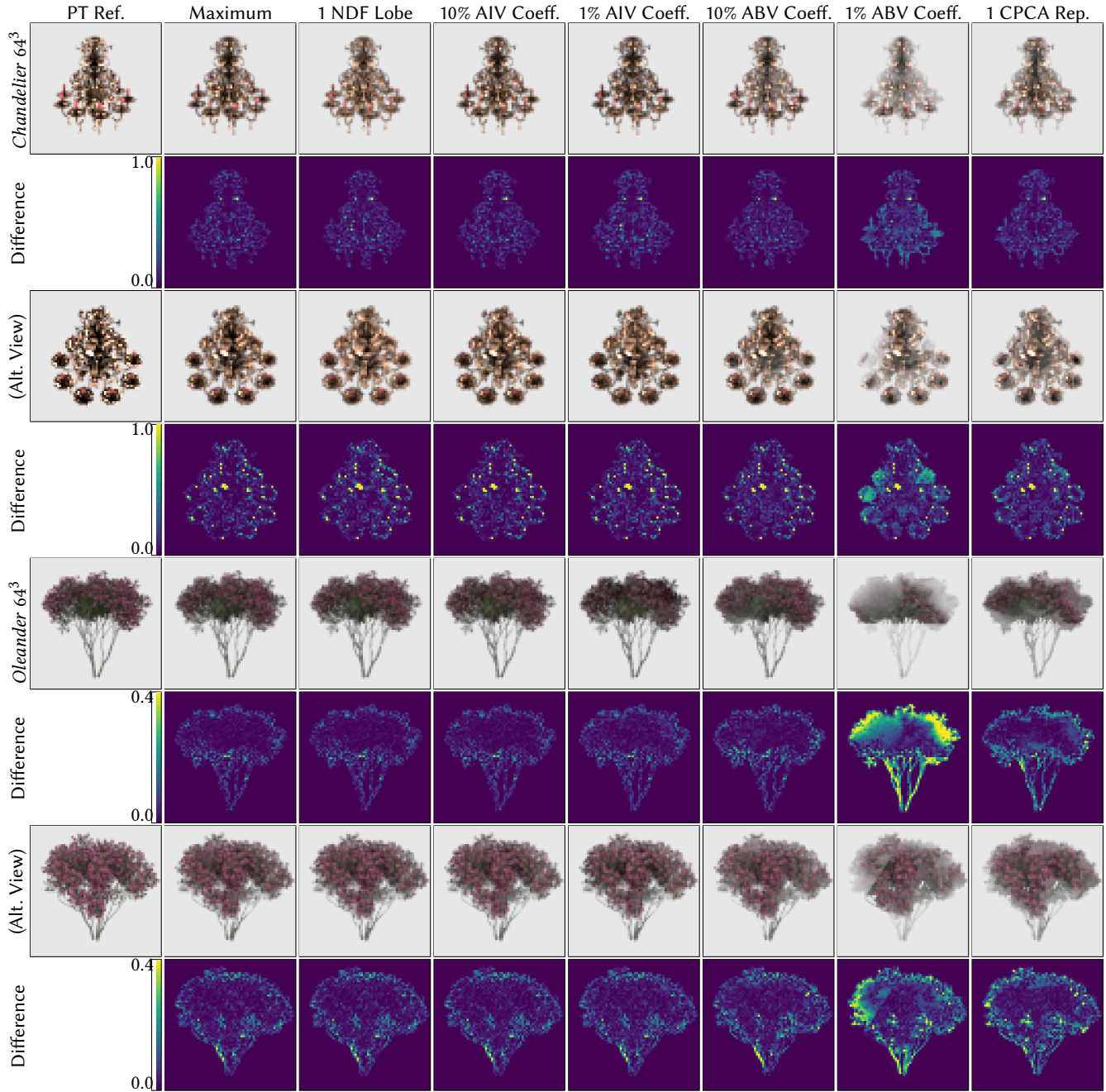
Fig. 18. Ablation study analyzing the impact of parameters on appearance and cost. All images are rendered using 1024 samples per pixel. Difference images between each configuration and the corresponding reference are provided. Please refer to Table 4 for statistics.

Wenzel Jakob, Adam Arbree, Jonathan T. Moon, Kavita Bala, and Steve Marschner. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph.* 29, 4 (2010), 53:1–53:13. https://doi.org/10.1145/1778765.1778790

Wenzel Jakob, Milos Hasan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete stochastic microfacet models. *ACM Trans. Graph.* 33, 4 (2014), 115:1–115:10. https://doi.org/10.1145/2601097.2601186

Adrián Jarabo, Carlos Aliaga, and Diego Gutierrez. 2018. A radiative transfer framework for spatially-correlated materials. *ACM Trans. Graph.* 37, 4 (2018), 83. https://doi.org/10.1145/3197517.3201282

Adrián Jarabo, Hongzhi Wu, Julie Dorsey, Holly E. Rushmeier, and Diego Gutierrez. 2014. Effects of Approximate Filtering on the Appearance of Bidirectional Texture Functions. *IEEE Trans. Vis. Comput. Graph.* 20, 6 (2014), 880–892. https://doi.org/10.1109/TVCG.2014.2312016

Jorge Jiménez, Xianchun Wu, Angelo Pesce, and Adrian Jarabo. 2016. Practical real-time strategies for accurate indirect occlusion. *SIGGRAPH 2016 Courses: Physically Based Shading in Theory and Practice* (2016).

James T. Kajiya and Timothy L. Kay. 1989. Rendering fur with three dimensional textures. *ACM Trans. Graph.* (1989), 271–280. https://doi.org/10.1145/74333.74361

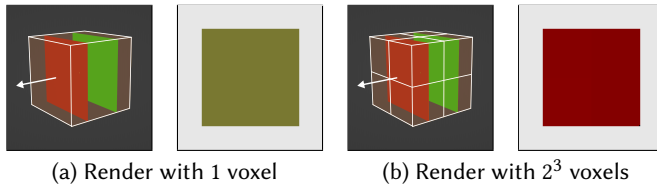(a) Render with 1 voxel   (b) Render with 2^3 voxels

Fig. 19. (a) Due the separate visibility approximation (Eq. 6), our method cannot handle correlation between visibility and material in a single voxel. The render averages the contribution from the red and green materials. (b) However, this limitation is mitigated as the voxelization resolution increases.

Viktor Kämpe, Erik Sintorn, and Ulf Assarsson. 2013. High resolution sparse voxel DAGs. *ACM Trans. Graph.* 32, 4 (2013), 101:1–101:13. https://doi.org/10.1145/2461912.2462024

Anton S. Kaplanyan, Stephan Hill, Anjul Patney, and Aaron E. Lefohn. 2016. Filtering distributions of normals for shading antialiasing. In *Proceedings of High Performance Graphics, HPG 2016, Dublin, Ireland, June 20-22, 2016*, David Luebke and Steven Molnar (Eds.). Eurographics Association, 151–162. https://doi.org/10.2312/hpg.20161201

Brian Karis, Rune Stubbe, and Graham Wihlidal. 2021. A Deep Dive into Nanite Virtualized Geometry. In *ACM SIGGRAPH 2021 Talks*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)* 42, 4 (2023), 1–14.

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2021. NeuMIP: multi-resolution neural materials. *ACM Trans. Graph.* 40, 4 (2021), 175:1–175:13. https://doi.org/10.1145/3450626.3459795

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2022. Rendering Neural Materials on Curved Surfaces, Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann (Eds.). *ACM Trans. Graph.*, 9:1–9:9. https://doi.org/10.1145/3528233.3530721

Samuli Laine and Tero Karras. 2010. Efficient sparse voxel octrees. In *Proceedings of the 2010 Symposium on Interactive 3D Graphics, SI3D 2010, February 19-21, 2010, Washington, DC, USA*, Daniel G. Aliaga, Manuel M. Oliveira, Amitabh Varshney, and Chris Wyman (Eds.). ACM, 55–63. https://doi.org/10.1145/1730804.1730814

Xinguo Liu, Peter-Pike J. Sloan, Heung-Yeung Shum, and John Snyder. 2004. All-Frequency Precomputed Radiance Transfer for Glossy Objects. In *Proceedings of the 15th Eurographics Workshop on Rendering Techniques, Norköping, Sweden, June 21-23, 2004*, Alexander Keller and Henrik Wann Jensen (Eds.). Eurographics Association, 337–344. https://doi.org/10.2312/EGWR/EGSR04/337-344

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhöfer, Yaser Sheikh, and Jason M. Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.* 40, 4 (2021), 59:1–59:13. https://doi.org/10.1145/3450626.3459863

Guillaume Loubet and Fabrice Neyret. 2017. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Comput. Graph. Forum* 36, 2 (2017), 431–442. https://doi.org/10.1111/cgf.13138

Guillaume Loubet and Fabrice Neyret. 2018. A new microflake model with microscopic self-shadowing for accurate volume downsampling. *Comput. Graph. Forum* 37, 2, 111–121. https://doi.org/10.1111/cgf.13346

David Luebke, Martin Reddy, Jonathan D Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. 2003. *Level of detail for 3D graphics*. Morgan Kaufmann.

Linjie Lyu, Ayush Tewari, Thomas Leimkühler, Marc Habermann, and Christian Theobalt. 2022. Neural Radiance Transfer Fields for Relightable Novel-View Synthesis with Global Illumination. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII (Lecture Notes in Computer Science, Vol. 13677)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 153–169. https://doi.org/10.1007/978-3-031-19790-1_10

Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: adaptive coordinate networks for neural scene representation. *ACM Trans. Graph.* 40, 4 (2021), 58:1–58:13. https://doi.org/10.1145/3450626.3459785

Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus H. Gross, and Wojciech Jarosz. 2015. Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph.* 34, 4 (2015), 49:1–49:13. https://doi.org/10.1145/2766949

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12346)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael

Frahm (Eds.). Springer, 405–421. https://doi.org/10.1007/978-3-030-58452-8_24

Jonathan T. Moon, Bruce Walter, and Steve Marschner. 2008. Efficient multiple scattering in hair using spherical harmonics. Vol. 27. 31. https://doi.org/10.1145/1360612.1360630

Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner. 2007. Rendering Discrete Random Media Using Precomputed Scattering Solutions. In *Proceedings of the Eurographics Symposium on Rendering Techniques, Grenoble, France, 2007*, Jan Kautz and Sumanta N. Pattanaik (Eds.). Eurographics Association, 231–242. https://doi.org/10.2312/EGWR/EGSR07/231-242

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* 41, 4 (2022), 102:1–102:15. https://doi.org/10.1145/3528223.3530127

Thomas Müller, Marios Papas, Markus H. Gross, Wojciech Jarosz, and Jan Novák. 2016. Efficient rendering of heterogeneous polydisperse granular media. *ACM Trans. Graph.* 35, 6 (2016), 168:1–168:14. https://doi.org/10.1145/2980179.2982429

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3 (2013), 27:1–27:22. https://doi.org/10.1145/2487228.2487235

Ken Museth. 2021. NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation. *ACM Trans. Graph.*, 1:1–1:2. https://doi.org/10.1145/3450623.3464653

Fabrice Neyret. 1998. Modeling, Animating, and Rendering Complex Scenes Using Volumetric Textures. *IEEE Trans. Vis. Comput. Graph.* 4, 1 (1998), 55–70. https://doi.org/10.1109/2945.675652

Marc Olano and Dan Baker. 2010. LEAN mapping. In *Proceedings of the 2010 Symposium on Interactive 3D Graphics, SI3D 2010, February 19-21, 2010, Washington, DC, USA*, Daniel G. Aliaga, Manuel M. Oliveira, Amitabh Varshney, and Chris Wyman (Eds.). ACM, 181–188. https://doi.org/10.1145/1730804.1730834

Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 165–174. https://doi.org/10.1109/CVPR.2019.00025

Jiangfeng She, Xiaoyan Gu, Junzhong Tan, Ming Tong, and Chaofan Wang. 2019. An appearance-preserving simplification method for complex 3D building models. *Trans. GIS* 23, 2 (2019), 275–293. https://doi.org/10.1111/tgis.12518

Peter Shirley and Kenneth Chiu. 1997. A Low Distortion Map Between Disk and Square. *J. Graphics, GPU, & Game Tools* 2, 3 (1997), 45–52. https://doi.org/10.1080/10867651.1997.10487479

Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. 2003. Clustered Principal Components for Precomputed Radiance Transfer. *ACM Trans. Graph.* 22, 3 (jul 2003), 382–391. https://doi.org/10.1145/882262.882281

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles T. Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 11358–11367. https://doi.org/10.1109/CVPR46437.2021.01120

Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul P. Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik. 2022. Advances in Neural Rendering. *Comput. Graph. Forum* 41, 2 (2022), 703–735. https://doi.org/10.1111/cgf.14507

Michael J. Todd. 2016. *Minimum volume ellipsoids - theory and algorithms*. MOS-SIAM Series on Optimization, Vol. 23. SIAM.

Michael Toksvig. 2005. Mipmapping Normal Maps. *J. Graph. Tools* 10, 3 (2005), 65–71. https://doi.org/10.1080/2151237X.2005.10129203

Delio Vicini, Wenzel Jakob, and Anton Kaplanyan. 2021. A non-exponential transmittance model for volumetric scene representations. *ACM Trans. Graph.* 40, 4 (2021), 136:1–136:16. https://doi.org/10.1145/3450626.3459815

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: a kernel framework for efficient CPU ray tracing. *ACM Trans. Graph.* 33, 4 (2014), 143:1–143:8. https://doi.org/10.1145/2601097.2601199

Beibei Wang, Milos Hasan, Nicolas Holzschuch, and Ling-Qi Yan. 2020. Example-Based Microstructure Rendering with Constant Storage. *ACM Trans. Graph.* 39, 5 (2020), 162:1–162:12. https://doi.org/10.1145/3406836

Jiaping Wang, Peiran Ren, Minmin Gong, John M. Snyder, and Baining Guo. 2009. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph.* 28, 5 (2009), 133. https://doi.org/10.1145/1618452.1618479

Philippe Weier, Tobias Zirr, Anton Kaplanyan, Ling-Qi Yan, and Philipp Slusallek. 2023. Neural Prefiltering for Correlation-Aware Levels of Detail. *ACM Trans. Graph.*

Hongzhi Wu, Julie Dorsey, and Holly E. Rushmeier. 2011. Physically-based interactive bi-scale material design. *ACM Trans. Graph.* 30, 6 (2011), 145. https://doi.org/10.1145/2070781.2024179

Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. 2019. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM*

Trans. Graph. 38, 4 (2019), 137:1–137:14. https://doi.org/10.1145/3306346.3322936

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural Fields in Visual Computing and Beyond. *Comput. Graph. Forum* 41, 2 (2022), 641–676. https://doi.org/10.1111/cgf.14505

Chao Xu, Rui Wang, Shuang Zhao, and Hujun Bao. 2017. Real-Time Linear BRDF MIP-Mapping. *Comput. Graph. Forum* 36, 4 (2017), 27–34. https://doi.org/10.1111/cgf.13221

Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. 2013. Anisotropic spherical Gaussians. *ACM Trans. Graph.* 32, 6 (2013), 209:1–209:11. https://doi.org/10.1145/2508363.2508386

Ling-Qi Yan, Milos Hasan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Trans. Graph.* 33, 4 (2014), 116:1–116:9. https://doi.org/10.1145/2601097.2601155

Ling-Qi Yan, Milos Hasan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-normal distributions for efficient rendering of specular microstructure. *ACM Trans. Graph.* 35, 4 (2016), 56:1–56:9. https://doi.org/10.1145/2897824.2925915

Cheng Zhang and Shuang Zhao. 2020. Multi-Scale Appearance Modeling of Granular Materials with Continuously Varying Grain Properties. In *31st Eurographics Symposium on Rendering, EGSR 2020 - Digital Library Only Track, London, UK, June 29 - July 3, 2020*, Carsten Dachsbacher and Matt Pharr (Eds.). Eurographics Association, 25–37. https://doi.org/10.2312/sr.20201134

Shuang Zhao, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph.* 35, 6 (2016), 166:1–166:11. https://doi.org/10.1145/2980179.2980228

Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. 2011. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph.* 30, 4 (2011), 44. https://doi.org/10.1145/2010324.1964939

Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. 2012. Structure-aware synthesis for predictive woven fabric appearance. *ACM Trans. Graph.* 31, 4 (2012), 75:1–75:10. https://doi.org/10.1145/2185520.2185571

Tobias Zirr and Anton S. Kaplanyan. 2016. Real-time rendering of procedural multiscale materials, Chris Wyman and Cem Yuksel (Eds.). *ACM Trans. Graph.*, 139–148. https://doi.org/10.1145/2856400.2856409