Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets

Guillaume Loubet, Fabrice Neyret

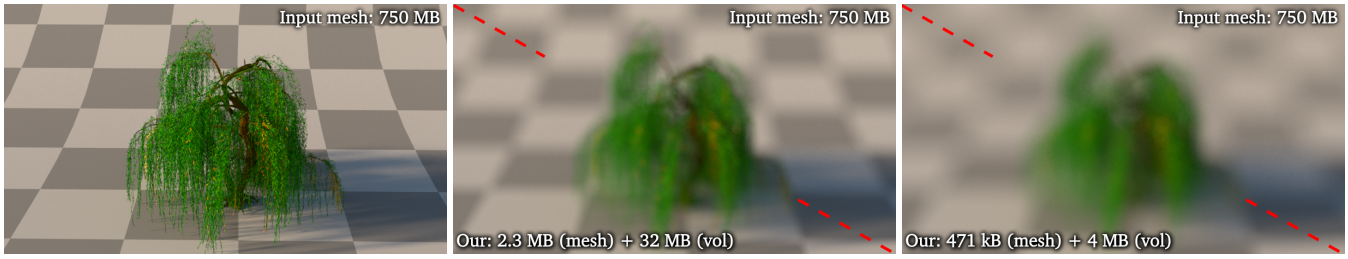
EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets

Guillaume Loubet† and Fabrice Neyret†

INRIA, Univ. Grenoble Alpes/LJK, CNRS/LJK



(a) Input mesh (b) LoD 1 ╲ Reference (c) LoD 2 ╲ Reference

Figure 1: *A 3D model pre-filtered with our method.* ***(b,c)****: Top right, reference image. Bottom left, our results. Our LoDs use meshes for representing macroscopic surfaces and a heterogeneous participating medium to approximate sub-resolution geometry. This approach allows for preserving the appearance of complex geometry across scales while reducing memory usage drastically. These images have been rendered with 256 samples per pixel, a thin lens camera model and a* `sunsky` *emitter in Mitsuba [Jak10].*

Abstract

*We address the problem of constructing appearance-preserving level of details (LoDs) of complex 3D models such as trees. We propose a hybrid method that combines the strengths of mesh and volume representations. Our main idea is to separate macroscopic (*i.e. *larger than the target spatial resolution) and microscopic (sub-resolution) surfaces at each scale and to treat them differently, because meshes are very efficient at representing macroscopic surfaces while sub-resolution geometry benefits from volumetric approximations. We introduce a new algorithm that detects the macroscopic surfaces of a mesh for a given resolution. We simplify these surfaces with edge collapses and we provide a method for pre-filtering their normal distributions and albedos. To approximate microscopic details, we use a heterogeneous microflake participating medium and we introduce a new artifact-free voxelization algorithm that preserves local occlusion. Thanks to our macroscopic surface analysis, our algorithm is fully automatic and it generates seamless LoDs at arbitrarily coarse resolutions for a wide range of 3D models.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Raytracing—

1. Introduction

Accurately pre-filtering 3D models is a major practical problem for rendering very detailed scenes, in particular for off-line rendering production environments such as film industry. Huge amounts of details can be prohibitive due to loading time, costly ray intersections and incoherent access to shading parameters [ENSB13]. Rendering becomes inefficient or even intractable when the entire scene does not fit into memory. Level of detail (LoD) techniques are often mandatory [PMA14, SJM16] not only during lookdev and lighting workflows but also for final renderings. Simplifying complex 3D models can drastically reduce loading time, memory usage and noise, but automatic and accurate pre-filtering of arbitrary assets at arbitrary scale remains an open problem.

Surface-based LoD methods fail to preserve the appearance of complex 3D models because sub-resolution intricate details of-

† guillaume.loubet@inria.fr & fabrice.neyret@inria.fr

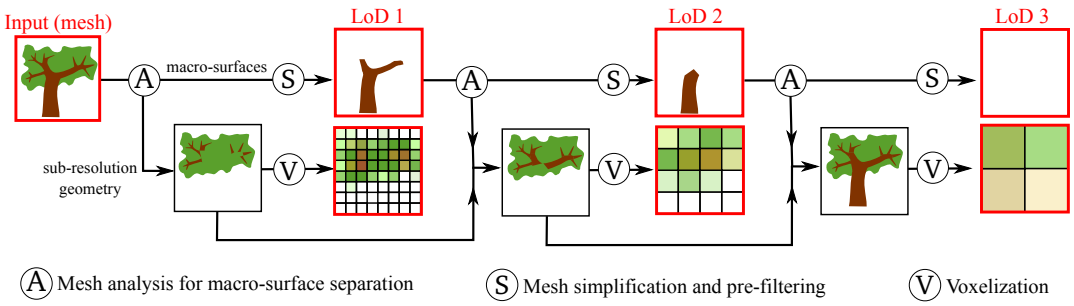


Figure 2: *Our pipeline: starting from an input textured mesh, we analyze and separate macro-surfaces from the rest of the geometry using our separation criterion (Ⓐ, Section 3). Macro-surfaces are pre-filtered by edge collapse until they match the target resolution (Ⓢ, Section 4). The rest of the geometry,* i.e. *sub-resolution occluders, is voxelized at each scale (Ⓥ, Section 5). Each output LoDs consists of a mesh containing pre-filtering macro-surfaces and voxels approximating sub-resolution occluders. They have to be rendered jointly. Our LoDs allow for seamless transitions (Section 6.2).*

(a) Input mesh (b) Isotropic medium approx. (c) Anisotropic medium approx. (d) Anisotropic medium approx. Density ×20

Figure 3: *Several approximations of a mesh containing macro-surfaces (with respect to voxel size) using heterogeneous participating media. Volumes are rendered with volumetric path tracing in Mitsuba. All voxelization strategies fail to preserve both watertightness **(b,c)** and local occlusion **(d)**. **(a)**: Input mesh. **(b)**: Isotropic participating medium approximation with preservation of the mean local occlusion (Section 5). **(c)**: Using an anisotropic participating medium model (microflakes). **(d)**: Increasing density preserves watertightness but leads to incorrect local occlusion and silhouettes.*

ten have a semi-transparent macroscopic appearance that cannot be easily pre-filtered using surface representations. On the other hand, volumetric models are neither efficient nor accurate for representing macroscopic surfaces: voxelized macro-surfaces are either too transparent or too thick (Fig. 3). Complex 3D models such as trees often show both macro-surfaces and sub-resolution details at a given scale, meaning that both surface-only and volume-only LoDs fail to accurately preserve their appearance.

Our work aims at combining the strength of volume and mesh representations. We represent macro-surfaces with meshes to preserve large scale watertightness and accurate silhouettes. We approximate sub-resolution geometry with a heterogeneous participating medium model in order to drastically reduce memory usage while preserving local probabilities of occlusion. Recently, the ellipsoid/SGGX normal distribution has been used as a microfacet normal distribution in a BRDF model [DWMG15] and as a microflake normal distribution [HDCD15] in the microflake participating media model [JAM*10]. We rely on this representation for pre-filtering normal distributions and preserving the appearance across scales.

Our pipeline is illustrated in Fig. 2. We generate LoDs at different resolutions. Seamless transitions can be achieved by interpolating meshes, surface reflectance parameters and volumetric data as described in Section 6.2. The choice of LoD for rendering may depend on the distance to the camera, the camera depth of field (Fig. 1) or other criteria such as the visibility (*e.g.* if the asset is out of frustum). In each LoD, the voxel size and edge lengths in the mesh match the target spatial resolution. We divide the target resolution by a factor 2 between two successive LoDs, so that 8 voxels in one LoD correspond to one voxel in the coarser LoD. Meshes and volumes are intended to be rendered jointly, *i.e.* meshes lie *inside* a heterogeneous participating medium.

Our work includes the following main contributions:

- We introduce a method for separating macro-surfaces from sub-resolution geometry at a given scale based on mesh analysis (Section 3).
- We provide an algorithm for pre-filtering albedos and microfacet distributions on meshes simplified with edge collapses (Section 4).
- We provide an artifact-free voxelization algorithm based on ray casting that preserves local occlusion (Section 5).

Thanks to our macro-surface analysis, our method automatically makes the best use of meshes and volumes at each scale without requiring *a priori* assumptions about the geometry. It can pre-filter a large class of 3D models with spatially varying details or mixtures of heterogeneous details (Fig. 1).

2. Related work

Our method relies on mesh-based LoDs, normal distribution pre-filtering and volumetric approximations of small elements.

2.1. Surface pre-filtering

Mesh-based geometry simplification. Mesh-based methods for LoDs have a long history in computer graphics and divide in two

main categories: subdivision methods and mesh simplification algorithms.

The former use low-resolution 3D models and manage LoDs through tessellation of coarse polygons and displacement, often encoded in texture patches [JH12, DHI*13, LBG16]. These methods cannot be used for pre-filtering the geometry at arbitrary scale because whole patches can themselves be sub-resolution.

Mesh simplification methods start with arbitrary meshes and reduce the mesh complexity by deleting or merging elements such as vertices. Luebke at al. proposed a comprehensive survey [LWC*02]. Mesh simplification methods can generate accurate approximations as long as surfaces are large enough compared to the resolution, and they fail to preserve correct occlusion in the case of sub-resolution intricate details. For instance, mesh simplification algorithms cannot be used for pre-filtering trees at very coarse scales. We follow ideas of Hoppe et al. and use edge collapses for pre-filtering macroscopic surfaces in our method, because it supports seamless transitions between LoDs [Hop96] and well-posed BRDF pre-filtering without ambiguities between front and back materials. It also preserves manifoldness, which allows for easier macro-surface analysis.

Other LoD representations for surface-like appearance. Hierarchical 3D data structures have been widely used for storing approximations of some input geometry. The underlying geometry in each voxel is often approximated with planar elements or similar representations [GM05, LK10, PES15]. For ray tracing applications, these methods face the problem of finding the best compromise between holes and shadowing artifacts due to disconnected primitives [WS05, LK10], and reflectance pre-filtering is often ill-posed. Other methods resort to distance fields represented with voxels [HN12] with a loss of generality. Some methods have been called *hybrid* [GM05] because they combine several representations depending on the scale. They typically use meshes for the finest LoDs. Our method is also hybrid but we use a participating medium approximation for sub-resolution geometry instead of surface primitives.

Surface reflectance pre-filtering. Many mesh simplification methods focused on geometry but some methods addressed the problem of pre-filtering reflectance parameters. Some authors proposed to estimate attributes at vertices minimizing some attribute error functions [GH98, Hop99]. Unfortunately, these approaches cannot accurately pre-filter surface reflectance and do not adapt well to current reflectance models based on microfacet normal distributions. Cohen et al. provided an algorithm for simplifying uv-textured meshes [COM98], allowing for normal distribution pre-filtering in texture space [OB10, BN11, DHI*13]. This approach shares the limitations of subdivision methods and lacks from generality because it cannot pre-filter surfaces beyond the scale of texture patches.

Recently, Dong et al. proposed a new microfacet reflectance model based on the *ellipsoid normal distribution function* [DWMG15]. It can be seen as an off-centered GGX distribution [WMLT07]. This distribution is similar to the SGGX microflakes distribution [HDCD15]. It is parametrized by a 3×3 symmetric matrix. We rely on this work to pre-filter microfacet normal distributions during edge collapses without requiring surface parametrization (Section 4.3).

2.2. Approximations of aggregate details

Random pruning. Cook et al. proposed to simplify aggregate details (*i.e.* aggregations of similar small elements such as leaves) with random pruning [CHPR07]. Their method is only valid if detail statistics (*e.g.* size and color) are the same in the entire 3D model, and cannot be used for pre-filtering non-random geometry such as the sailing ship in Fig. 12.

Volumetric approximations of aggregate details. Volumetric models have been successfully used to approximate specific aggregations of details such as foliage [MKMW97], fibers [MWM08, SKZ11, ZJMB11] or granular materials [MPH*15, MPG*16]. Unfortunately, these techniques are not intended to pre-filter arbitrary sub-resolution geometry.

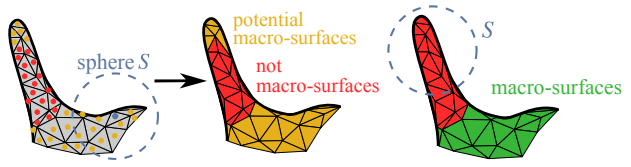
Occlusion pre-filtering. Crassin et al. [CNS*11] used voxels with anisotropic opacity for down-sampling volumetric data but they do not address the problem of estimating the correct opacity given some input geometry. Lacewell et al. [LBBS08] addressed this problem by measuring local occlusion due to small occluders, which is also our approach. They showed that pre-filtering occlusion can significantly reduce noise in rendered images. Their method is not fully automatic since the user must specify the scale at which the geometry can be approximated with a semi-transparent coarse occluder. On the contrary, our method automatically separates macro-surfaces from sub-resolution geometry (Section 3). We also pre-filter reflectance parameters and we propose an aliasing-free voxelization algorithm.

2.3. Microflake participating media

Jakob et al. proposed a physically-based anisotropic participating media framework based on a microflake model [JAM*10]. This model relies on normal distributions and projected areas of microflakes. We use this model in our work for approximating sub-resolution geometry and we directly estimate microflake normal distributions from microfacet normal distributions.

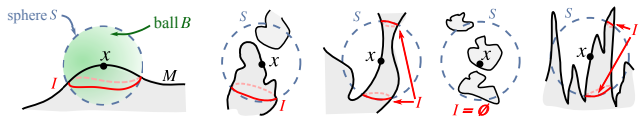
Heitz et al. revisited Neyret's micro-ellipsoids model [Ney95, Ney98] and proposed the SGGX microflakes distribution [HDCD15]. They showed that a mix of SGGX distribution can be approximated with a single SGGX distribution by linear combination of SGGX parameters (this approximation is accurate enough if the mix of SGGX is indeed a SGGX-like distribution). We rely on this work to efficiently pre-filter normal distributions.

Recently, Zhao et al. [ZWDR16] proposed a method for down-sampling volumetric data based on the microflake model with the SGGX distribution. We address a different problem: our inputs are meshes and one of our goals is to preserve the appearance of macro-surfaces that cannot be accurately pre-filtered using volumetric representations. We also address the problem of voxelizing some geometry. Contrary to them, we do not directly down-sample our volumetric data since we voxelize sub-resolution geometry at each scale (Fig. 2).



(a) Step 1 (b) Step 2

Figure 4: *Our two steps macro-surface analysis.* ***(a)****: Detecting potential macro-surfaces (orange). Our criterion uses spheres whose radius is equal to the target resolution and whose centers are on the surface (Section 3.1).* ***(b)****: Keeping only large connected surfaces (green). As the upper part of the bump is smaller that S, it is analyzed as sub-resolution geometry.*



(a) (b) (c) (d) (e)

Figure 5: *Several geometries including non-ambiguous macro-surfaces* ***(a)****, sub-resolution geometry* ***(c,d)*** *and ambiguous cases* ***(b,e)****. Our macro-surface analysis at a point x on a surface M depends on the number of connected components of I, defined in Section 3.1 as the set of points $p \in (M \cap S)$ that are connected to x inside the ball B.* ***(a,b)****: I has a single component, our criterion detects a potential macro-surface at x.* ***(c,d,e)****: I is empty or has more than one component. The point x is analyzed as not belonging to a macro-surface.*

3. Automatic macro-surface analysis

A key element of our approach is the separation of macro-surfaces from sub-resolution geometry. It corresponds to the step Ⓐ in our pipeline (Fig. 2). The difficulty comes from the lack of a clear definition of what is a macro-surface. Examples of ambiguous cases include highly curved and chaotic surfaces as shown in Fig. 5b and 5e.

We propose an algorithm that robustly detects unambiguous macro-surfaces such as large smooth surfaces compared to the resolution (Fig. 5a), and that separates them from sub-resolution disconnected (Fig. 5d), cylinder-like and ribbon-like elements (Fig. 5c).

We propose a method based on local mesh analysis. We leverage the fact that in most 3D models, watertight macro-surfaces behaviors are modeled with a single connected surface. Our method looks for large connected macro-surfaces in a given neighborhood.

3.1. Our mesh analysis

The input of our analysis is a triangle mesh whose edges are smaller than the target resolution. We introduce a method for detecting macro-surfaces given a target resolution. In a first step, we find triangles that could belong to macro-surfaces. In a second step, we only keep connected surfaces that are large enough compared to the resolution (Fig. 4).

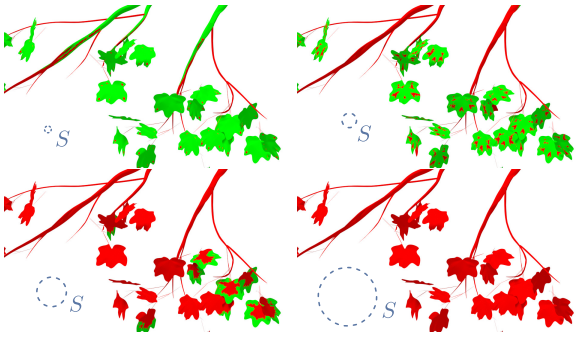


Figure 6: *Results of our mesh-based macro-surface analysis with four different resolutions: blue circles represent the size of sphere S (Section 3.1) whose radius is equal to the target spatial resolution in the LoD. Green triangles are triangles analyzed as belonging to macro-surfaces by our algorithm. Top left: leaves and big twigs are macro-surfaces compared to the target resolution. Bottom right: the geometry has no surface wider than the target resolution.*

Step 1. Let x be a point on the mesh M and r the target resolution. Let S be the sphere of radius r centered in x, and B the closed ball bounded by S (Fig. 5a). We call I the set of points $p \in (M \cap S)$ that are connected to x in $M \cap B$. We say that x belongs to a potential macro-surface if I has *one single connected component*. Let's look at what happens in two simple cases:

- I is empty for all points on any disconnected geometry that is small compared to S (Fig. 5d). Points on such small elements are analyzed as not belonging to macro-surfaces.
- For all points on cylinders whose radius is small compared to S, I has two connected components, as shown in Fig. 5c, meaning that such cylinders are analyzed as sub-resolution geometry.

In practice, we compute the topology of I one time per triangle with x being the triangle barycenter.

Step 2. We remove *potential macro-surfaces* given by the first step that are sub-resolution (Fig. 4b, upper part of the bump). We analyze each connected piece of potential macro-surfaces and keep only connected surfaces whose bounding sphere is larger than S.

Results of our mesh analysis are shown in Fig. 6. Pseudo code of our algorithm is given in Appendix A and implementation details can be found in Section 6.1.

3.2. Discussion

A limitation of our approach is that it only detects macro-surfaces due to connected and smooth enough meshes. It cannot detect large scale watertightness due to disconnected elements or very rough surfaces (Fig. 5e, 14c). We also implemented an algorithm based on measurement of local directional occlusion but detecting large-scale watertight macro-surface with local measurement lacks robustness.

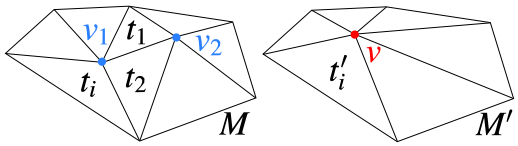


Figure 7: *Our notations for one edge collapse. v_1 and v_2 are merged into a new vertex v. Triangles t_1 and t_2 are collapsed. Each other triangle t_i in M corresponds to a new triangle t'_i in M'.*

4. Mesh simplification and surface pre-filtering

We now address the problem of simplifying meshes representing macro-surfaces (Ⓢ in Fig. 2) while pre-filtering their reflectance. We propose a pre-filtering method that does not require any surface parametrization, unlike texture-based pre-filtering methods for example. We pre-filter microfacet normal distributions in world space and simplify macro-surfaces at arbitrary scale without being constrained by texture patches.

Again, we consider triangular meshes with edges smaller than the target resolution. We simplify the geometry using edge collapses until edge lengths match the target resolution. As triangles in our LoDs are never larger than the target resolution, we use per-triangle reflectance parameters (*i.e.* albedos and microfacet distributions). After each edge collapse, we update neighboring triangles with pre-filtered reflectance parameters.

We first describe our mesh simplification method and then we present our algorithm for pre-filtering albedos and microfacet distributions at each edge collapse.

4.1. Edge collapse simplification

It has two key ingredients: the way the algorithm builds a priority list of edges to be collapsed, and the strategy for vertex placement at each collapse.

Current vertex placement strategies based on the quadric error metric (or QEM) [GH97, Hop99] lack accuracy because they tend to shrink the geometry. We observed in our experiments that volume preserving methods such as the one proposed by Lindstrom et al. [LT98] performs better but lack robustness for thin geometry. We discuss this problem and we propose a new vertex placement method in our supplemental material.

Unlike budget-based and fidelity-based strategies [LWC*02], our method is resolution-based and we only apply edge collapse on macro-surfaces. We found that in this context, building the priority list for the collapses is less critical than the choice of the vertex placement strategy. We used a strategy based on the standard QEM provided by the OpenMesh library [BSBK02, Ope] and forbid collapses of edges larger than the target resolution.

4.2. Reflectance model

We use the microfacet-based BRDF model proposed by Dong et al. [DWMG15]. We store the 6 coefficients of the SGGX distribution (called the ellipsoid NDF by Dong et al.) for each triangle as well as albedos for diffuse and specular reflections. The specular term is given by

$$f_s(\omega_i,\omega_o,\mathbf{S}) = \frac{D(\mathbf{h},\mathbf{S})\; G_2(\omega_i,\omega_o,\mathbf{h},\mathbf{S})\; F(\omega_i\cdot\mathbf{h})}{4|\omega_i\cdot\mathbf{n}||\omega_o\cdot\mathbf{n}|} \tag{4.1}$$

with $\mathbf{S}$ the SGGX matrix, D the microfacet SGGX normal distribution, G_2 the masking-shadowing term, F the Fresnel term, $\mathbf{h}$ the half vector and $\mathbf{n}$ the normal. Analytic forms and sampling procedures can be found in the supplemental material provided by Dong et al. [DWMG15]. For diffuse microfacets, we follow the formulation and sampling procedure proposed by Heitz et al. [HD15]. It is given by

$$f_d(\omega_i,\omega_o,\mathbf{S}) = \frac{1}{\pi}\frac{1}{|\omega_o\cdot\mathbf{n}|}\int_\Omega \langle\omega_o,\omega\rangle \frac{G_2(\omega_i,\omega_o,\mathbf{h},\mathbf{S})}{G_1(\omega_i,\mathbf{h},\mathbf{S})} D_{\omega_i}(\omega,\mathbf{S})d\omega \tag{4.2}$$

with D_{ω_i} the visible normal distribution in direction ω_i and G_1 the masking term. $\langle\ ,\ \rangle$ is the clamped dot product. Our per-triangle BRDF consists of specular and diffuse terms:

$$f_i(\omega_i,\omega_o) = s_i f_s(\omega_i,\omega_o,\mathbf{S_i}) + d_i f_d(\omega_i,\omega_o,\mathbf{S_i}) \tag{4.3}$$

with s_i the triangle specular albedo and d_i the diffuse albedo. $\mathbf{S_i}$ is the SGGX matrix.

Given an input textured mesh, per-triangle attributes can be retrieved with texture sampling. GGX roughness directly translates to the SGGX representation. Our method could be easily extended to specular and diffuse transmission terms [WMLT07, DWMG15].

4.3. BRDF pre-filtering during edge collapses

In this section, we show how to use the linear pre-filtering method proposed by Heitz et al. [HDCD15] in the edge collapse framework. At each collapse, we linearly combine microfacet normal distributions and albedos of neighboring triangles in order to find new parameters for the new triangles. We address the problem of choosing appropriate weights for these linear combinations so that the surface appearance is preserved during the mesh simplification.

Assuming that triangles are sub-resolution, we want to pre-filter per-triangle parameters so that the mean appearance of M is preserved in M' (Fig. 7) at each edge collapse. We consider the planar case, *i.e.* when triangles adjacent to the collapsed edge all lie in the same plane, and we derive a pre-filtering method based on the area of triangles. Non planar cases are discussed later.

Notations. Let's consider the collapse of the edge joining two vertices v_1 and v_2 as illustrated in Fig. 7. We call the new vertex v. We assume for clarity that neither v_1 nor v_2 are border vertices in this explanation. Our results extend straightforwardly to the general case. Let t_i be the N triangles adjacent to v_1 and v_2 ($i\in[1,N]$) such that t_1 and t_2 are the collapsed faces. Let t'_i ($i\in[3,N]$) be the faces after the collapse such that each t_i corresponds to t'_i. We call M the set of triangles t_i and M' the set of triangles t'_i (Fig. 7). Let A_i and A'_i be the area of triangles t_i and t'_i. We set $A'_1=0$ and $A'_2=0$ since t_1 and t_2 are collapsed. It is convenient to use the notation w_i for the area of each triangle t_i divided by the area of M:

$$\forall i\in[1,N],\quad w_i = \frac{A_i}{\sum_k A_k}. \tag{4.4}$$

Similarly, we define

$$\forall i \in [1,N],\ \ w'_i = \frac{A'_i}{\sum_k A'_k}. \tag{4.5}$$

By definition, we have $\sum w_i = 1$ and $\sum w'_i = 1$, $w'_1 = w'_2 = 0$.

Preserving the mean radiance. In the planar case, and assuming a locally uniform lighting, the mean radiance of surface M in direction ω_o is given by:

$$L_{mean}(\omega_o) = \sum_{i\in[1,N]} w_i \int_\Omega L(\omega) f_i(\omega,\omega_o)\langle\omega,\mathbf{n}\rangle\, d\omega. \tag{4.6}$$

We want to find BRDFs f'_i for each triangle t'_i so that the mean radiance is preserved after the collapse, meaning $L_{mean}(\omega_o) = L'_{mean}(\omega_o)$ with

$$L'_{mean}(\omega_o) = \sum_{i\in[3,N]} w'_i \int_\Omega L(\omega) f'_i(\omega,\omega_o)\langle\omega,\mathbf{n}\rangle\, d\omega. \tag{4.7}$$

Our approach. Our method consists in writing each BRDF f'_i as a linear combination of BRDFs f_i:

$$\forall j \in [3,N],\ \ f'_j = \sum_{i\in[1,N]} w_i^j f_i \quad \text{with } \forall j,\ \sum_i w_i^j = 1 \tag{4.8}$$

The mean radiance of M' in the planar case is given by

$$\begin{aligned} L'_{mean}(\omega_o) &= \sum_j w'_j \int_\Omega L(\omega) \sum_i w_i^j f_i(\omega,\omega_o)\langle\omega,\mathbf{n}\rangle\, d\omega \\ &= \sum_i \sum_j w'_j w_i^j \int_\Omega L(\omega) f_i(\omega,\omega_o)\langle\omega,\mathbf{n}\rangle\, d\omega. \end{aligned} \tag{4.9}$$

Then, our problem is to find appropriate weights w_i^j so that $L'_{mean}(\omega_o) = L_{mean}(\omega_o)$. Thus, we want to ensure that

$$\forall i \in [1,N],\ \ \sum_j w'_j w_i^j = w_i. \tag{4.10}$$

There is a simple and naïve solution, which consists in choosing $w_i^j = w_i\ \forall j$. This solution results in uselessly blurred surfaces: all the parameters are averaged in the neighborhood of the collapsed edge, even if the collapsed edge is actually very small compared to neighboring triangles. Instead, we propose a simple heuristic for the choice of weights w_i^j: we maximize weights w_i^i (*i.e.* the weight of f_i in the new triangle t'_i) in order to avoid over-blurring, and so that BRDFs f'_i preserve the mean radiance in each direction. Finally, we use these weights for linearly pre-filter albedos and SGGX matrices per triangle as described later.

Let $G = \{i \in [1..N],\ w_i < w'_i\}$ be the set of triangles whose area increase with the edge collapse, and $L = \{i \in [1..N],\ w_i \geq w'_i\}$ the other ones.

Our weights are the following:

$$\begin{cases} w_i^i = 1 & \forall i \in L \\ w_j^i = 0 & \forall i \in L \text{ and } i \neq j \\ w_i^i = \dfrac{w_i}{w'_i} & \forall i \in G \\ w_j^i = (1 - \dfrac{w_i}{w'_i}) c_j & \forall i \in G \text{ and } i \neq j \\ \text{with } c_j = \dfrac{w_j - w'_j}{\sum_{i\in L} w_i - w'_i} \end{cases} \tag{4.11}$$

A proof that this solution satisfies Eq. 4.10 can be found in Appendix B.

Pre-filtering. Once all the weights w_i^j have been computed, we pre-filter parameters assuming that albedos and normal distributions are not correlated and we linearly pre-filter SGGX coefficients as follows:

$$\begin{aligned} \forall j \in [3,N],\quad f'_j &= \sum_{i\in[1,N]} w_i^j f_i \\ &= \sum_i w_i^j \Big(s_i f_s(\omega_i,\omega_o,\mathbf{S_i}) + d_i f_d(\omega_i,\omega_o,\mathbf{S_i}) \Big) \\ &\simeq s'_j f_s(\omega_i,\omega_o,\mathbf{S'_i}) + d'_j f_d(\omega_i,\omega_o,\mathbf{S'_i}) \end{aligned} \tag{4.12}$$

with $s'_j = \sum_i w_i^j s_i$ the mean specular albedo, $d'_j = \sum_i w_i^j d_i$ the mean diffuse albedo and $S'_i = \sum_i w_i^j \mathbf{S_i}$ the filtered normal distribution.

4.4. Discussion

Our pre-filtering method is fast because it relies on linear combinations and because weights are given by very simple formulas that only depend on the area of each triangle. In our implementation, it takes around 7 μs per collapse including the pre-filtering itself, which is small compared to the cost of the vertex placement (around 170 μs in our implementation). Our method can pre-filter anisotropic roughness due to small surface details as shown in Fig. 8.

We focus on the planar case for the derivation of weights and use this algorithm for all edge collapses. Our method is less accurate on non planar cases, because the emitted radiance in direction ω_o depends on cosine weights $\langle\omega_o,\mathbf{n_i}\rangle$ for each triangle with normal $\mathbf{n_i}$, and also on view dependent visibility terms taking into account masking between triangles, leading to a much more difficult problem.

Our method is also limited by the use of a single SGGX lobe: it cannot pre-filter surfaces with strong microscopic correlations between the geometry and albedos [HSRG07, HN12] or normal distributions that are very different from SGGX lobes. More accurate solutions could be computed using multi-lobe SGGX distributions similarly to Zhao et al. [ZWDR16] in order to handle more complex appearances and avoid accumulation of errors during the process, leading to overly rough normal distributions.

5. Voxelization

We now address the problem of converting the micro-geometry separated from macro-surfaces into a volumetric representation. This

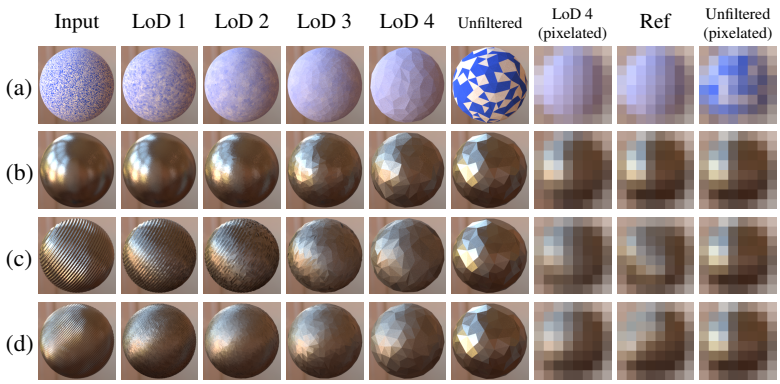


Figure 8: *Meshes simplified with edge collapses and pre-filtered with our method (Section 4.3).* ***(a)****: Sphere with diffuse microfacets.* ***(b)****: Sphere with specular microfacets. Coarse LoDs have rougher materials since our pre-filtering is done in world space and takes into account surface curvature.* ***(c,d)****: Spheres with specular materials and grooves leading to anisotropic normal distributions in coarse LoDs.*

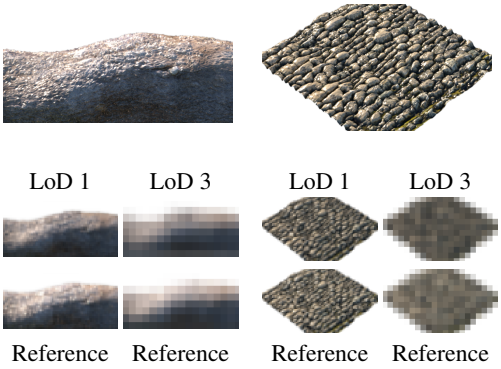


Figure 9: *Macro-surface pre-filtering on two assets with diffuse and specular microfacets. Left: a tree trunk. Right: Stone pavers. Rendered in Mistuba (*`sunsky` *emitter). Pixel sizes are chosen so that simplified triangles are sub-pixel.*

corresponds to step Ⓥ in Fig. 2. We use heterogeneous participating media with the microflakes model [JAM*10]. Our data for each voxel consists of a density parameter, 6 SGGX coefficients representing the microflake distribution [HDCD15], specular and diffuse albedos. We first discuss the problem of object-space aliasing in voxelizations and then we propose an algorithm based on ray casting that avoids voxelization artifacts.

5.1. Background

Voxelization often refers to binary voxelizations with empty or non-empty voxels. We address a different problem: we want to estimate voxel parameters such as density to approximate some input geometry. Previous approaches measured occlusion due to the geometry contained in cubes or in cuboids [LBBS08, Pan14]. This leads to object-space aliasing artifacts in voxelized data as already noted in [WK93]. An example is given in Fig. 11. Artifacts can be avoided if the voxelization process satisfies the Nyquist sampling criterion. A low-pass filter must be used, meaning that a voxel should not represent a cubical domain but a blurry and large enough overlapping domain around the voxel center.

Our approach consists in casting rays around each voxel center and looking for intersection with the geometry in the voxel neighborhood. The main idea of our voxelization algorithm is that we give the same weight to each ray but we sample rays with probability density functions (*pdf*) that act like low-pass filters and prevent object-space aliasing. More specifically, we choose ray origins neither on a cube nor uniformly inside a cube. Instead, we use a smooth *pdf* around a given voxel. We sample a random direction uniformly on the spherical domain for each ray in order to prevent directional artifacts.

5.2. Occlusion estimation with ray casting

We cast rays with length ray_l: we ignore intersections farther than ray_l from ray origins. We only take into account the first intersection with the geometry, if any. Therefore, we estimate the probability of occlusion in a voxel neighborhood along a distance ray_l. Long rays due to a large value ray_l would tend to over-blur the voxelization, while small rays (small ray_l) would only measure how much the geometry fills the space. In our implementation, we set ray_l to the voxel size.

When rays intersect the geometry, we store albedos and normal distributions, and we pre-filter them for each voxel. An estimation of the probability of occlusion in the voxel neighborhood over a distance ray_l in given by

$$P_{occ} = \frac{\text{nbHits}}{\text{nbRays}} \tag{5.1}$$

where nbRays is the number of rays casted for this voxel and nbHits is the number of rays that intersected some geometry at a distance smaller that ray_l. We use P_{occ} to retrieve a density parameter for the voxel in Section 5.4.

5.3. Artifact-free voxelization

We now propose a sampling algorithm for casting rays around voxels in order to avoid object-space aliasing. Let v_i be the center of a voxel i, and D_i the probability density function used to randomly sample ray origins for that voxel (Fig. 10). We want to know how much (*i.e.* with what probability) each point $x \in \mathbb{R}^3$ can be hit by a ray during the voxelization, because we need this probability to vary smoothly in space in order to prevent object-space aliasing. We introduce the function P_i that gives, for each point $x \in \mathbb{R}^3$, the probability that a ray whose origin is sampled using the *pdf* D_i intersects a spherical occluder with radius *rad* centered in x. This function P_i should be smooth enough: it is the low-pass filter of the voxelization. Moreover, given that there are N_v voxels, we want the function P_{all} defined by

$$P_{all}(x) = \frac{1}{N_v} \sum_{voxels\ i} P_i(x)$$

to be constant $\forall x \in \mathbb{R}^3$, *i.e.* all points in space have the same probability of being intersected by rays during the entire voxelization (this is also for preventing aliasing).

Probability of intersecting an occluder. Let's first consider a single ray with a given origin and a random direction. This ray can intersect the geometry around the ray origin with some probability that depends on the distance to the ray origin. Let P_r be the radial function that gives the probability that a ray hit a spherical occluder of radius *rad*, given that the occluder is at a distance d from the ray origin. Intuitively, $P_r(d)$ decreases as long as d increases because the solid angle of the spherical occluder decreases. The solid angle of a cone with apex angle θ is given by $2\pi(1-\cos(\theta))$. The solid angle of a spherical occluder is the one of a cone with apex

$$\theta = \arccos\left(\frac{\sqrt{d^2 - rad^2}}{d}\right)$$

where d is the distance between the ray origin and the occluder center. For $d \geq rad$ and $d \leq ray_l$, we have

$$P_r(d) = \frac{1}{2}\left(1 - \sqrt{1 - \frac{rad^2}{d^2}}\right)$$

which tends to $\frac{rad^2}{4d^2}$ when *rad* becomes very small compared to d.

Sampling strategies. Given D_i (the *pdf* for ray origins) and P_r (the probability of intersection depending on the distance from a ray origin), we can write the probability P_i of being intersected at x during the voxelization of voxel i as

$$P_i(x) = \int_{\mathbb{R}^3} P_r(\mathrm{dist}(x,p))\, D_i(p)\, \mathrm{d}p.$$

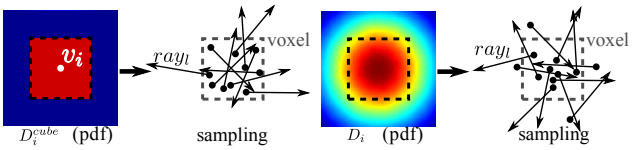
The probability of being intersected at x during the entire voxelization is the weighted sum of $P_i(x)$ over all the voxels:

$$\begin{aligned} P_{all}(x) &= \frac{1}{N_v}\sum_i \int_{\mathbb{R}^3} P_r(dist(x,p))\, D_i(p)\, \mathrm{d}p \\ &= \frac{1}{N_v}\int_{\mathbb{R}^3} P_r(dist(x,p)) \left(\sum_i D_i(p)\right) \mathrm{d}p \\ &= \frac{1}{N_v}\int_{\mathbb{R}^3} P_r(dist(0,p)) \left(\sum_i D_i(p+x)\right) \mathrm{d}p. \end{aligned}$$

Now P_{all} is constant over space if $\sum_i D_i$ is constant. A simple solution is to choose a cubic uniform probability density function:

$$D_i^{cube}(x) = \begin{cases} \dfrac{1}{\mathrm{voxelSize}^3} & \text{if } \|x - v_i\|_1 \leq \mathrm{voxelSize} \\ 0 & \text{otherwise} \end{cases}$$

where $\|\ \|_1$ is the ℓ_1 norm. Moreover, any convolution of a solution with another probability density functions is also a solution. Smooth functions P_i can be achieved using $D_i = D_i^{cube} * D^{smooth}$, with D^{smooth} any smooth probability density function. In our implementation, we used a 3D Gaussian function with standard deviation $\sigma = 0.6$ voxelSize (example (c) in Fig. 11). Pseudo-code for our sampling method can be found in Algorithm 1.



(a) Sampling ray origins uniformly in a cube.

(b) Sampling ray origins using a smoother *pdf*.

Figure 10: Sampling strategies for ray origins using different *pdf*. We sample a random direction for each ray in order to avoid artifacts.

Algorithm 1: *Our sampling algorithm for ray origins (Section 5.3). We sample origins in a cube around the voxel and then use a 3D normal distribution in order to avoid object-space aliasing.*

```
nbHits = 0;
rayL = voxelSize;
mu = (0,0,0);
sigma = 0.6×voxelSize ;
for i = 1 to nbRays do
    rayD = SampleDir ();
    rayO = voxelCenter
         + (SampleUnitCube () - (1/2, 1/2, 1/2)) × voxelSize
         + SampleNormalDistrib (mu, sigma);
    if Intersect (rayO, rayD, rayL) then
        nbHits ++;
    end
end
```

5.4. Estimating density parameters

At a given voxel, Eq. 5.1 estimates the local probability occlusion of rays with length ray_l in the voxel neighborhood. We need to estimate the voxel density parameter ρ so that attenuation equals P_{occ} for a distance ray_l in the participating medium. In the microflakes framework, the unit projected area of microflakes in direction ω is given by

$$\sigma_u(\omega) = \int_\Omega |\omega \cdot m| D(m)\, \mathrm{d}m,$$

with D the microflake normal distribution function. We look for the density parameter ρ that ensures

$$1 - \frac{1}{4\pi}\int_\Omega e^{-\rho\sigma_u(\omega) ray_l}\, \mathrm{d}\omega = P_{occ}$$

In general, there is no closed-form for parameter ρ. One can show that the function f defined as

$$f(\rho) = \frac{1}{4\pi}\int_\Omega e^{-\rho\sigma_u(\omega) ray_l}\, \mathrm{d}\omega$$

is a strictly decreasing function and has C^∞ continuity, which makes numerical methods based on 1D gradient descents very efficient at estimating ρ. It requires estimations of f and $\frac{\partial f}{\partial \rho}$. In our implementation, we estimate them using numerical integration.

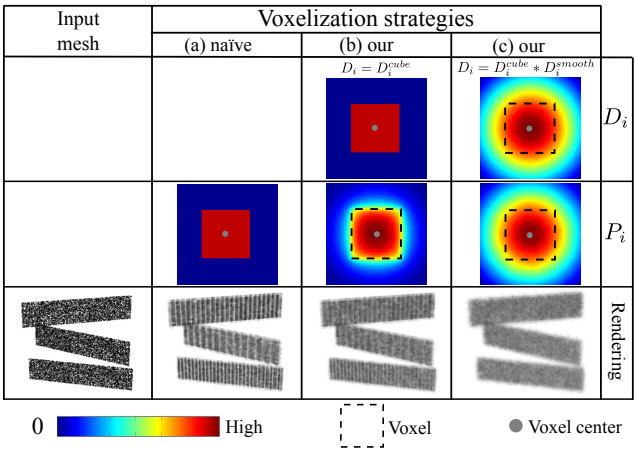


Figure 11: *Effect of voxelization strategies on object-space aliasing. The naïve voxelization* ***(a)*** *consists in measuring the occlusion of geometry in a small cube around the voxel center. Our voxelization algorithm can prevent object-space aliasing and acts as a low-pass filter* ***(b,c)****. Images represent slices of the 3D probability density functions D_i for sampling ray origins (Section 5.3) and slices of 3D functions P_i giving for each point the probability of being intersected by a ray.*

6. Implementation and results

6.1. Macro-surface analysis

We implemented our analysis with the OpenMesh library [BSBK02, Ope] and use the half-edge data structure for efficient mesh local exploration.

The cost of our analysis for a given point x depends on how many triangles are connected to x in B. Our implementation is expensive if the mesh is very detailed compared to S because a lot of triangles will have to be considered. However, in our filtering pipeline (Fig. 2) the average triangle size increases at each LoD and their edge lengths are approximately $\frac{r}{2}$. Moreover, only macro-surfaces from the previous LoD have to be analyzed. Consequently, our implementation is efficient for our pipeline. Parallel implementations are straightforward since each triangle analysis is independent. In our implementation, the mean cost of the analysis per triangle is around 15 μs if the mesh comes from the previous LoD (the analysis of the input mesh highly depends on its resolution compared to the first target resolution).

6.2. Seamless transitions between LoDs

We give some implementation details concerning seamless transitions between between successive LoDs, each one consisting of voxels and a mesh (Fig. 2) with per-triangle reflectance attributes.

Macro-surfaces. Given the macro-surfaces of LoD_N, the set of edge collapses that leads to LoD_{N+1} defines a mapping between vertices of LoD_N and vertices of LoD_{N+1} [Hop96]. We interpolate vertex positions in order to achieve smooth transitions between the two LoDs.

For each triangle in LoD_{N+1} there is a corresponding triangle in LoD_N. We smoothly interpolate their albedos and SGGX coefficients during the transition between LoD_N and LoD_{N+1}. As we use SGGX coefficients in world space, specular reflections stay consistent during the transition even if triangles are changing.

From meshes to volume. Triangles of LoD_N that are voxelized in LoD_{N+1} (Fig. 2) must smoothly fade out during the transition, while voxel densities smoothly increase. We simply use an opacity coefficient for each voxelized triangle that goes from 1 to 0 during the transition.

Interpolating volumes. 8 voxels in LoD_N correspond to 1 voxel in LoD_{N+1}. In our implementation we linearly interpolate densities, albedos and SGGX coefficients of voxels in LoD_N with the corresponding voxels in LoD_{N+1}. We do not use spatial interpolation between voxels of the same LoD because our voxelization algorithm prevents high frequencies in the volumetric data and because it facilitates seamless transitions.

6.3. Other implementation details

Voxelization. We used Jakob's Mistuba renderer [Jak10] for casting rays during our voxelization (Section 5). We casted 200 rays for each non-empty voxel and stored 16 bytes: 4 for the density parameter (a *float*), 6 for the SGGX coefficients (1 byte per coefficient), 3 for the specular albedo and 3 for the diffuse albedo (1 byte per color channel).

Rendering. We also used Mitsuba for rendering our LoDs. We implemented plugins for the diffuse and specular SGGX phase functions based on the code provided by Heitz et al. [HDCD15]. We modified the core of the renderer in order to render microflake media with different specular and diffuse albedos. We also implemented the BRDF model of Dong at al. [DWMG15] for specular and diffuse microfacets based on the work of Heitz et al. [HD15].

Efficiency. Pre-filtering computation times can be found in Fig. 13. Our macro-surface pre-filtering was run on a single core, while voxelization was using 8 hyperthreaded cores on a Intel Xeon E5-2609. For the first LoD of the weeping willow model the total computation time was 20 minutes, which includes 2 minutes for loading the input mesh and textures, 5 minutes for the macro-surface analysis and 4 minutes for the mesh simplification and pre-filtering. The remaining time corresponds to various other tasks such as preparing and writing the outputs.

6.4. Results

We tested our method on complex surfaces (Fig. 8, 9) and complex 3D assets including trees and a sailing ship model with tiny ropes and wood elements (Fig. 12). Seamless transitions between our LoDs are demonstrated in our supplemental video. Fig. 1 shows our LoDs rendered in a scene with different diaphragm apertures. Examples of limitations are shown in Fig. 14. As discussed in Fig. 3 and Section 3.2, they are mainly due to the limitations of our macro-surface analysis and the inappropriate use of volumes for representing almost surface-like appearance.

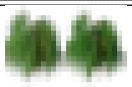
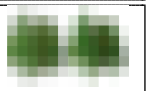
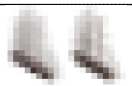
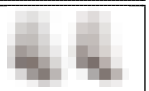
Input mesh	Res. 512^3 / ref	Res. 256^3 / ref	Res. 128^3 / ref	Res. 64^3 / ref	Res. 32^3 / ref	Res. 16^3 / ref
						
						
Error: 0%	0.90%	1.7%	2.1%	2.8%	4.4%	8.3%
Mesh: 692 MB Mat: 60.4 MB	17.7 MB 2.7 MB	1.9 MB 405 kB	379 kB 91 kB	60.2 kB 16.4 kB	0 0	0 0
Vol: none Sparsity: none Full grid: none	84 MB 97% 2 GB	18 MB 94% 256 MB	4 MB 90% 32 MB	827 kB 83% 4 MB	139 kB 76% 512 kB	33 kB 57% 64 kB
						
						
Error: 0%	1.3%	0.94%	2.8%	0.63%	1.3%	7.3%
Mesh: 43 MB Mat: 2.2 MB	2.7 MB 456 kB	654 kB 134 kB	176 kB 44.2 kB	52.4 kB 15.9 kB	6.7 kB 3.7 kB	1.3 kB 2.0 kB
Vol: none Sparsity: none	6 MB 99.8%	1.5 MB 99.5%	389 kB 99.0%	131 kB 97.3%	30 kB 95.0%	8 kB 89.0%
						
						
Error: 0%	-2.4%	-1.35%	0.75%	2.86%	5.11%	7.97%
Mesh: 78 MB Mat: 9.2 MB	35 MB 5 MB	1.4 MB 332 kB	235 kB 61 kB	24 kB 8.6 kB	2.8 kB 2.8 kB	0 0
Vol: none Sparsity: none	45 MB 98.4%	13 MB 95.7%	23.4 MB 92.3%	623 kB 87.2%	246 kB 59.6%	26 kB 65.2%

Figure 12: *Our LoDs rendered with a volumetric path tracer and the* `sunsky` *emitter in Mitsuba. All assets have diffuse and specular microfacets. The size of volumetric data takes into account the sparsity. We added bytes for voxel indices in the grid (6 bytes for the LoD* 512^3 *and 3 for the other resolutions, per voxel). Occlusion errors were measured in image space.*

	Willow	**Vessel**	**Maple**
Macro-surface	40 mn	3 mn	3 mn
Vox. 512^3	16 mn	12 mn	13 mn
Vox. 256^3	4 mn	2 mn	2 mn

Figure 13: *Pre-filtering times for the 3 assets shown in Fig. 12. Computation times for pre-filtering macro-surfaces include the loading time for input meshes and textures, our macro-surface analysis, the mesh simplification step with our pre-filtering algorithm and other tasks like writing the new meshes and textures.*

(a) Diffuse spheres with different sizes.

(b) A diffuse plane with holes.

(c) Tree trunk and grass (very rough surface).

Figure 14: *Our hybrid LoDs for 3 input meshes (top). Volumetric parts are shown in green (left columns).* ***(b)*** *and* ***(c)*** *show limitations of our method: our mesh analysis cannot detect surface-like appearance when the surface is disconnected, has holes* ***(b)*** *or is not smooth enough,* e.g. *because of the grass under the trunk in* ***(c)****. Our volumetric approximation is too transparent in such cases.*

7. Conclusion and future work

The key idea of our work is that a large variety of 3D assets can be automatically pre-filtered at arbitrary scale if macro-surfaces and sub-resolution details are pre-filtered separately, using the most appropriate representations. We showed that using both meshes and volumes allows for preserving large scale watertightness and occlusion of complex geometry, unlike surface-only and volume-only methods. We relied on reflectance models that support linear pre-filtering and that are memory efficient, and used the SGGX distribution for surfaces and volumes in the same framework. Our LoDs have low memory footprints and could be used for rendering very complex scenes that wouldn't have fitted into memory, with little loss in quality.

Our hybrid approach could be extended to other surface reflectance models and volumetric models in order to balance differently accuracy and efficiency. In particular, we think that multi-lobe approaches in the spirit of Zhao et al. [ZWDR16] could be adapted to our work in order to support a wider range of macroscopic appearances. Our approach would also benefit from more complex volumetric models with separated anisotropic occlusion and phase functions. We believe that the need for accurate pre-filtering will inspire future work on models for microscopic shadowing and view-dependent appearance, as well as new tools to link and unify surface and volumetric representations.

Acknowledgement

We would like to thank Antoine Bouthors for very helpful discussions, as well as Eric Heitz and Joëlle Thollot for their feedbacks on the paper. We also thank *Edson* for his sailing vessel 3D model and Olivier Ffrench[†] for the mossy stone wall.

References

[BN11] BRUNETON E., NEYRET F.: A survey of non-linear pre-filtering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics* (2011). 3

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh - a generic and efficient polygon mesh data structure. In *OpenSG Symposium* (2002). 5, 9

[CHPR07] COOK R. L., HALSTEAD J., PLANCK M., RYU D.: Stochastic simplification of aggregate detail. *ACM Trans. Graph.* (2007). 3

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)* (2011). 3

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *Proceedings of SIGGRAPH '98* (1998), pp. 115–122. 3

[DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 211:1–211:11. 3

[DWMG15] DONG Z., WALTER B., MARSCHNER S., GREENBERG D. P.: Predicting appearance from measured microgeometry of metal surfaces. *ACM Trans. Graph. 35*, 1 (Dec. 2015), 9:1–9:13. 2, 3, 5, 9

[ENSB13] EISENACHER C., NICHOLS G., SELLE A., BURLEY B.: Sorted deferred shading for production path tracing. In *Proceedings of EGSR '13* (2013), pp. 125–132. 1

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH '97* (1997), pp. 209–216. 5

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the IEEE VIS '98* (1998), pp. 263–269. 3

[GM05] GOBBETTI E., MARTON F.: Far Voxels – a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms. *ACM Trans. Graph.* (2005), 878–885. 3

[HD15] HEITZ E., DUPUY J.: *Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation*. Tech. rep., 2015. URL: https://eheitzresearch.wordpress.com/research/. 5, 9

[HDCD15] HEITZ E., DUPUY J., CRASSIN C., DACHSBACHER C.: The SGGX microflake distribution. *ACM Trans. Graph. 34*, 4 (July 2015), 48:1–48:11. 2, 3, 5, 7, 9

[HN12] HEITZ E., NEYRET F.: Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. In *EGGH-HPG'12 - Eurographics conference on High Performance Graphics* (2012), pp. 125–134. 3, 6

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of SIGGRAPH '96* (1996), ACM, pp. 99–108. 3, 9

[Hop99] HOPPE H.: New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of IEEE VIS '99* (1999), pp. –. 3, 5

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. *ACM Trans. Graph.* (2007). 6

[Jak10] JAKOB W.: Mitsuba renderer, 2010. URL: http://www.mitsuba-renderer.org. 1, 9

[JAM*10] JAKOB W., ARBREE A., MOON J. T., BALA K., MARSCHNER S.: A radiative transfer framework for rendering materials with anisotropic structure. *ACM Trans. Graph. 29*, 4 (July 2010), 53:1–53:13. 2, 3, 7

[JH12] JANG H., HAN J.: Feature-preserving displacement mapping with graphics processing unit (GPU) tessellation. *Comput. Graph. Forum* (2012), 1880–1894. 3

[LBBS08] LACEWELL D., BURLEY B., BOULOS S., SHIRLEY P.: Raytracing prefiltered occlusion for aggregate geometry. In *IEEE Symposium on Interactive Raytracing* (2008). 3, 7

[LBG16] LAMBERT T., BÉNARD P., GUENNEBAUD G.: Multi-Resolution Meshes for Feature-Aware Hardware Tessellation. *Computer Graphics Forum* (2016). 3

[LK10] LAINE S., KARRAS T.: Efficient sparse voxel octrees. In *Proceedings of I3D '10* (2010), pp. 55–63. 3

[LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. In *IEEE VIS '98* (1998), pp. 279–286. 5

[LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002. 3, 5

[MKMW97] MAX N., KEATING B., MOBLEY C., WU E.-H.: Plane-parallel radiance transport for global illumination in vegetation. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97* (1997), pp. 239–250. 3

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

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

[†] http://www.virtual-lands-3d.com

[MWM08] MOON J. T., WALTER B., MARSCHNER S.: Efficient multiple scattering in hair using spherical harmonics. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 31:1–31:7. 3

[Ney95] NEYRET F.: A General and Multiscale Model for Volumetric Textures. In *Graphics Interface* (1995), pp. 83–91. 3

[Ney98] NEYRET F.: Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (Jan. 1998), 55–70. 3

[OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the I3D '10* (2010), ACM, pp. 181–188. 3

[Ope] URL: https://www.openmesh.org/. 5, 9

[Pan14] PANTELEEV A.: Practical real-time voxel-based global illumination for current GPUs. Presented at SIGGRAPH '14 - Best of GTC talks, 2014. 7

[PES15] PRUS M., EISENACHER C., STAMMINGER M.: Level-of-Detail for Production-Scale Path Tracing. In *Vision, Modeling and Visualization* (2015). 3

[PMA14] PALMER S., MAURER E., ADAMS M.: Using sparse voxel octrees in a level-of-detail pipeline for Rio 2. In *ACM SIGGRAPH 2014 Talks* (2014), ACM, pp. 70:1–70:1. 1

[SJM16] SCHWANK A., JAMES C. J., MICILOTTA T.: The trees of the Jungle Book. In *ACM SIGGRAPH 2016 Talks* (2016), pp. 21:1–21:2. 1

[SKZ11] SCHRÖDER K., KLEIN R., ZINKE A.: A volumetric approach to predictive rendering of fabrics. In *Proceedings of EGSR '11* (2011), pp. 1277–1286. 3

[WK93] WANG S. W., KAUFMAN A. E.: Volume sampled voxelization of geometric primitives. In *Proceedings of IEEE VIS '93* (1993), pp. 78–84. 7

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of EGSR'07* (2007), pp. 195–206. 3, 5

[WS05] WALD I., SEIDEL H.-P.: Interactive ray tracing of point-based models. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics* (2005), pp. 9–16. 3

[ZJMB11] ZHAO S., JAKOB W., MARSCHNER S., BALA K.: Building volumetric appearance models of fabric using micro ct imaging. In *Proceedings of SIGGRAPH '11* (2011), pp. 44:1–44:10. 3

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

Appendix A: Computation of I

The topology of I, as defined in section 3.1, can be computed with a mesh exploration, starting from the face containing the point x. We make sure that only triangle connected to x *inside* S are explored. We do not need the exact shape of I but just its topology.

```
Function IsOnMacroSurface (inputTri, sphere)
    I = ∅;
    trisToExplore = inputTri;
    exploredTris = ∅;
    while not IsVoid (trisToExplore) do
        currentTri = PopFirst (trisToExplore);
        Explore (currentTri, I, exploredTris);
        Add (currentTri, exploredTris);
    end
    return HasSingleConnectedComponent (I);
End

Function Explore (currentTri, I, exploredTris,
                  trisToExplore, sphere)
    for e in edges of currentTri do
        if e intersect sphere or e is inside sphere then
            adjTri = triangle adjacent to currentTri at e;
            if adjTri is not in exploredTris and
               adjTri is not in trisToExplore then
                Add (trisToExplore, adjTri);
            end
        end
    end
    /* sphere ∩ currentTri could be 1, 2 or 3
       sections of a circle, or void. Only
       the ends of each part is needed,
       because we just need the topology of
       I.                                    */
    its = ComputeIntersections (currentTri, sphere);
    Add (its, I);
End
```

Appendix B: Pre-filtering the reflectance on edge-collapse

We check that contributions are preserved for each material (equation 4.10) by our method (Section 4.3). If $i \in G$,

$$\sum_j w_i^j w'_j = \frac{w_i}{w'_j} w'_j = w_i.$$

Else if $i \in L$,

$$\begin{aligned}
\sum_j w_i^j w'_j &= w'_i + \sum_{k\in G} w_i^k w'_k \\
&= w'_i + \sum_{k\in G} c_i (1 - \frac{w_k}{w'_k}) w'_k \\
&= w'_i + \sum_{k\in G} \frac{w_i - w'_i}{\sum_{l\in L} w_l - w'_l} (w'_k - w_k) \\
&= w'_i + (w_i - w'_i) \frac{\sum_{k\in G} w'_k - w_k}{\sum_{l\in L} w_l - w'_l}.
\end{aligned}$$

Observing that

$$\begin{aligned}
\sum_{k\in G} (w'_k - w_k) - \sum_{l\in L} (w_l - w'_l) &= \sum_{k\in G} (w'_k - w_k) + \sum_{l\in L} (w'_l - w_l) \\
&= \sum_i w'_i - \sum_j w_i \\
&= 1 - 1 \\
&= 0
\end{aligned}$$

we can write

$$\begin{aligned}
\sum_j w_i^j w'_j &= w'_i + (w_i - w'_i) \frac{\sum_{k\in G} w'_k - w_k}{\sum_{k\in G} w'_k - w_k} \\
&= w_i.
\end{aligned}$$

Therefore, our method satisfies equation 4.10.