Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees

Eric Heitz, Fabrice Neyret

To cite this version:

Eric Heitz, Fabrice Neyret. Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. EGGH-HPG'12 - Eurographics conference on High Performance Graphics, Jun 2012, Paris, France. pp.125-134, 10.2312/EGGH/HPG12/125-134 . hal-00704461

HAL Id: hal-00704461
https://inria.hal.science/hal-00704461

Submitted on 5 Jun 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

High Performance Graphics (2012)
C. Dachsbacher, J. Munkberg, and J. Pantaleoni (Editors)

Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees

Eric Heitz and Fabrice Neyret

INRIA Grenoble Rhône-Alpes and Laboratoire Jean Kuntzmann (Université de Grenoble and CNRS)

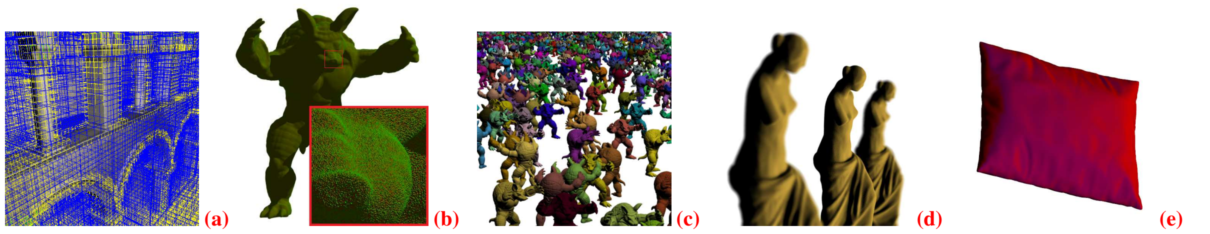

Figure 1: *Multiple surface details projecting within subpixels can produce complex shading effects that can be rendered in real-time with our pre-filtered SVO representation* (a). *Other real-time methods, such as MIPmapping, tend to neglect various correlation effects. Our method allows for correct filtering of color variations* (b), *anti-aliasing* (c), *and depth-of-field* (d), *without oversampling and with seamless transitions when zooming or defocusing. Moreover, our representation can be used directly to easily design light- and view-dependent materials* (e).

Abstract

Sparse Voxel Octrees (SVOs) represent efficiently complex geometry on current GPUs. Despite the fact that LoDs come naturally with octrees, interpolating and filtering SVOs are still issues in current approaches.

In this paper, we propose a representation for the appearance of a detailed surface with associated attributes stored within a voxel octree. We store macro- and micro-descriptors of the surface shape and associated attributes in each voxel. We represent the surface macroscopically with a signed distance field and we encode subvoxel micro-details with Gaussian descriptors of the surface and attributes within the voxel. Our voxels form a continuous field interpolated through space and scales, through which we cast conic rays. Within the ray marching steps, we compute the occlusion distribution produced by the macro-surface inside a pixel footprint, we use the micro-descriptors to reconstruct light- and view-dependent shading, and we combine fragments in an A-buffer way.

Our representation efficiently accounts for various subpixel effects. It can be continuously interpolated and filtered, it is scalable, and it allows for efficient depth-of-field. We illustrate the quality of these various effects by displaying surfaces at different scales, and we show that the timings per pixel are scale-independent.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Color, shading, shadowing, and texture—I.3.3 [Computer Graphics]: Antialiasing—

1 Introduction

With finite resolution screens, explicit shapes are not necessarily at the scale of a pixel footprint, and are even a drawback to render the pixel since either costly supersampling is required or disturbing aliasing will occur. Thus the interest for SVO in the gaming-oriented literature [Car08], representing complex local geometry only at the required scale, and greatly benefiting from the 3D hierarchical grid structure for efficient traversing. Still, subpixel features can impact a pixel value, and not only through the linear separable way that MIPmapping assumes: averaging maps is not sufficient to capture subpixel effects like non-linear attributes such as normals and roughness, and the contribution of a microsurface to a pixel is correlated to its visibility from the eye (masking) and the light (shadowing). This is even more obvious when an attribute such as color is correlated to depth within the local

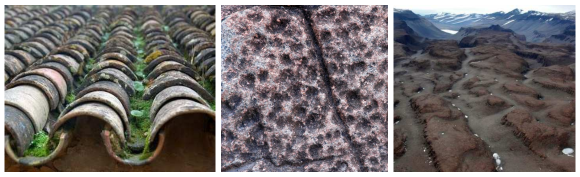

Figure 2: Real surfaces showing correlations of colors with depth. *Masking produces view-dependent color variations.*

© The Eurographics Association 2012.

heightfield, and thus with the visibility [BN11]. Numerous natural or human-made surfaces show such correlations of attributes with depth, from fabrics to barks, from soils and landscapes to structured surfaces, from fractured to rusty or aged layers (Figure 2).

2 Previous Work

2.1 Sparse Voxel Octree

John Carmack and *Id Software* [Car08] popularized voxel engines for displaying complex geometry in the gaming world. Because their voxels are not interpolated they look like Lego(TM) bricks at close view, and aliasing is high.

GigaVoxels [CNLE09] octrees store n^3 voxel bricks (in a 3D texture) at the nodes. They mimic volume density rendering and rely on 3D MIPmapping and texture interpolation. This representation makes interpolation possible through space and scale (including efficient soft shadows simulations and depth-of-field), but it ignores subvoxel correlation with visibility. Moreover, alpha or density is a crude approximation of subpixel occupancy when correlation between successive pixel fragments along a ray is high (see Figure 4), as it is on an object silhouette. This impacts the macroscopic color, lighting, and silhouette thickness.

Crassin et al. [CNS*11] encode light-view dependent energy and material (color + normal) in voxels, and filter normal sets as Normal Distribution Function (NDF) lobes used to control Phong exponent and cone aperture (i.e., the MIPmap level used to integrate incoming energy). But visibility is not accounted for during LoD pre-filtering, and the representation has preferred directions best adapted to architectural scenes.

Efficient Sparse Voxel Octrees (ESVO) [LK10a, LK10b, LK10c] are the first step toward surface (rather than volumetric) representation with subvoxel content. They explicitly represent the geometric occupancy within a voxel through an oriented slab engulfing surface variations, which allows view-dependent intersections. However, this approach has strong limitations (described in [LK10b] 3.2 and 3.3) concerning:

- Magnification: The representation makes spatial interpolation of the geometry impossible. Data are accessed in *nearest* mode which results in polyhedral aspect at close views and sharp aliased edges in the silhouettes.
- LoD: Quadrilinear interpolation of the geometry (between two levels in the octree) is not possible and produces popping artefacts at transitions. Moreover, the ESVO construction process is top-down which does not guarantee consistency of appearance between scales while top level appearance/BRDF should match the effect of bottom microstructures.
- Filtering: Aliasing occurs at distance even using several samples per pixel. Screen-space blurring has to be done to reduce these flaws.

A complete SVO model definitely needs to account for subgeometry distribution as in [LK10a], but on a continuous way (with interpolation) as in [CNLE09], taking into account effects on multiscale materials as in [CNS*11]. Correlation between surface attributes and visibility producing view-dependent effects should also be taken into account for many real-world surfaces. To make it possible on a generic way, we will reframe the rendering problem to be solve following a *differential cone tracing* formalism, i.e., the integration over the pixel using a ray and its footprint radius in the spirit of [Ige99].

2.2 BRDF

BRDFs have long be though of as the macroscopic light- and view-dependent appearance of a statistically defined microgeometry, including visibility effects [CT81, ON94]. However, the micro-surface is assumed as ideal (e.g.locally specular, or pure diffuse) and not varying in its attributes. Thus, the derivations should be reformulated to include local BRDF and colors (and any other attributes). Smith [Smi67] (see Eqs (4)-(6)) proposes a formulation of visibility similar to [CT81] but parameterized by depth within the heightfield. We will draw on this BRDF model since it is very adapted to account for depth-dependent attributes.

Fournier [Fou92] and Han et al. [HSRG07] propose to represent normal distributions as sum of lobes and to convolve BRDF and NDF. Note that in the scope of voxel storage and real-time rendering, the base memory footprint and calculation must be kept lightweight: decomposition in basis (e.g., Spherical Harmonics) is not affordable. Similarly to Olano et al. [OB10] we rely on a simple NDF lobe and a simple BRDF on microsurface to treat these attributes through our pre-integration.

2.3 Visibility and Correlation

Microfacet-based BRDF models above account for local light-view visibility with some simplifying assumptions. More complex cases (e.g.correlated visibilities) have been treated in other domains, from horizon maps to hotspot in satellite views of forests. See the survey on filtering non-linear effects on macro-surfaces [BN11].

Non-local visibility correlation along a cone ray has also to be considered: the occupancy distribution within a pixel fragment is equivalent to a transparency (or alpha) only in the case of non-correlation between the content of successive fragments. Classical volume rendering and GigaVoxels [CNLE09] assume random scatterers distribution within voxels, but this assumption cannot apply to SVOs since they represent opaque coherent objects: along a ray passing through the silhouette, all the subvoxel density is on the same side of the silhouette.

This has been studied for meshes in the scope of efficient anti-aliasing: two neighbor triangles are very correlated, and some manufactured objects have structured features prone to alignments. With his A-buffer algorithm [Car84], Carpenter proposed a representation of the subpixel occupancy through a compact bitmask (possibly stored in the bits of a single `int`). The bitmask of the various fragments along a ray are combined through logical operators to determine their contribution to the pixel. The 2D subpixel occupancy mask of a fragment is obtained from a pre-calculated mask table indexed by the edge rasterization within a pixel. Several variants, including vector masks, have been proposed. We inspire ourselves from this idea for combining fragments masks along a ray. Our fragments will correspond to traversed voxels, introducing a view-independent 3D vector mask to represent subvoxel occupancy. When marching along a ray, these will generate 2D bitmasks combined as for A-buffer.

3 Filtering Local Surface Appearance: Problem Study

Basic Problem and its Naive Solutions Let suppose that the BRDF ρ at location **x** can be expressed as a sum of elementary BRDFs [Pho75, LFTG97] weighted by attributes $a_i(\mathbf{x})$ (e.g.the specular, diffuse, and ambient terms scaled by color coefficients, in the Blinn-Phong model).

$$\rho(\mathbf{x},\mathbf{n}(\mathbf{x})) = \sum_i a_i(\mathbf{x})\,\rho_i(\mathbf{x},\mathbf{n}(\mathbf{x})) \tag{1}$$

a_i are usually expressed as RGB values.

The local illumination equation expresses the light intensity reflected by surface towards the observer as a function of the surface attributes. Radiance I perceived in direction **v** from surface A (meant to be a fragment of a given pixel's footprint) is:

$$I = \frac{\sum_i \int_A E_\mathbf{l}(\mathbf{x}) a_i(\mathbf{x}) \rho_i(\mathbf{v},\mathbf{l},\mathbf{x},\mathbf{n}(\mathbf{x}))\mathbf{ln}(\mathbf{x}) V(\mathbf{v},\mathbf{x}) V(\mathbf{l},\mathbf{x}) \mathbf{vn}(\mathbf{x})\,\mathrm{d}\mathbf{x}}{\int_A V(\mathbf{v},\mathbf{x})\ \mathbf{vn}(\mathbf{x})\ \mathrm{d}\mathbf{x}} \tag{2}$$

At point **x** of A, $\mathbf{vn}(\mathbf{x})$ and $\mathbf{ln}(\mathbf{x})$ are the clamped dot products between the surface normal and the eye and light directions, $V(\mathbf{v},\mathbf{x})$ and $V(\mathbf{l},\mathbf{x})$ designate the visibility values along the eye and light source, and $E_\mathbf{l}$ the entering radiance emitted by the environment from direction **l** (see [BN11] for more details).

Usual MIPmap-based mesh shading as well as 3D MIPmapping [CNLE09] assume separability of terms, averaging these attributes and approximating

$$I_i \approx \bar{E}_\mathbf{l}\ \bar{a}_i\ \rho_i(\mathbf{v},\mathbf{l},\bar{\mathbf{n}})\ \mathbf{l}\bar{\mathbf{n}}\ \bar{V}(\mathbf{l}) \tag{3}$$

where $\bar{E}_\mathbf{l} = \frac{\int_A E_\mathbf{l}(\mathbf{x})\ \mathrm{d}\mathbf{x}}{\int_A \mathrm{d}\mathbf{x}}$, $\bar{a}_i = \frac{\int_A a_i(\mathbf{x})\ \mathrm{d}\mathbf{x}}{\int_A \mathrm{d}\mathbf{x}}$, $\bar{\mathbf{n}} = \frac{\int_A \mathbf{n}(\mathbf{x})\ \mathrm{d}\mathbf{x}}{\int_A \mathrm{d}\mathbf{x}}$ and $\bar{V}(\mathbf{l}) = \frac{\int_A V(\mathbf{l},\mathbf{x})\ \mathrm{d}\mathbf{x}}{\int_A \mathrm{d}\mathbf{x}}$ are the surface mean values of the incoming radiance, the surface attributes, the normals, and the visibility from the light source. However, attributes $a_i(\mathbf{x})$ might be correlated with their visibilities $V(\mathbf{v},\mathbf{x})$ and $V(\mathbf{l},\mathbf{x})$ so their screen-wise mean value is not the mean of a_i, a constant, but a view-dependent function $\bar{a}_i(\mathbf{v},\mathbf{l})$. Also, applying the BRDF equation to the mean normal –$\rho(\mathbf{v},\mathbf{l},\bar{\mathbf{n}})$–

Symbol	Description
$\mathbf{x}$	local position on the surface A
$\mathbf{n}(\mathbf{x})$	local normal
$a(\mathbf{x})$	local attribute
$E_\mathbf{l}(\mathbf{x})$	incoming radiance from direction **l**
$\rho(\mathbf{v},\mathbf{l},\mathbf{x},\mathbf{n}(\mathbf{x}))$	local BRDF
$V(\mathbf{v},\mathbf{x})$	visibility of **x** from eye
$V(\mathbf{l},\mathbf{x})$	visibility of **x** from light source
$\bar{q}$	pixel-wise average of local quantity $q(\mathbf{x})$

does also not produce the correct result since a convolution with the normal distribution is missing: naively MIPmapping a specular bumpmap yields a specular macro-surface instead of diffuse.

Choosing a Microscopic Surface Model To account for the light-view-dependent effect of microgeometry, microfacet-based analytical BRDF models such as [CT81, ON94] reproduce the reflectance of surfaces of known (e.g., Gaussian) statistical properties. This principle could be used to filter macrogeometry as well, but the models above do not account for attribute variations along the surface: an attribute can be factored out the integral only if the BRDF is an affine function of it (e.g., Phong colors coefficients) and if attributes values are not correlated to their visibility. In practice, this hypothesis is often not valid since attributes are correlated to geometry by construction. Fortunately, for many real-world surfaces (Figure 2) attributes are simply correlated with depth h within the surface heightfield. In such a case, we can rely on Smith's formulation [Smi67] of micro-surface visibility which integration is parameterized over the depth within the heightfield, so that it is easy to revisit it adding an extra weight. For a surface where depth $h(\mathbf{x})$ and slopes (n_x, n_y) are two Gaussian random processes $\mathcal{N}(0,\sigma_h^2)$ and $\mathcal{N}(0,(\sigma_{n_x}^2,\sigma_{n_y}^2))$, then the probability of visibility $V(\mathbf{v},h)$ is given by [Smi67]:

$$V(\mathbf{v},h) = g(h)^{\Lambda(\mathbf{v})} \tag{4}$$

with

$$g(h) = 1 - \frac{1}{2}\,\mathrm{erfc}\left(\frac{h}{\sqrt{2}\sigma_h}\right) \tag{5}$$

$$\Lambda(\mathbf{v}) = \frac{1}{2}\left(\sqrt{\frac{2}{\pi}}\frac{\sigma_n(\mathbf{v})}{\mu(\mathbf{v})}e^{-\frac{\mu(\mathbf{v})^2}{2\sigma_n(\mathbf{v})^2}} - \mathrm{erfc}\left(\frac{\mu(\mathbf{v})}{\sqrt{2}\sigma_n(\mathbf{v})}\right)\right) \tag{6}$$

$\mu(\mathbf{v}) = \cot(\theta)$ where θ is the angle between the surface normal and the eye-direction, and $\sigma_n(\mathbf{v})$ is the distribution of the slopes in the projected direction of **v**. In our model, we use this formulation to compute the visibility of an attribute correlated to its depth in the surface.

4 Our General Rendering Framework

Formally, anti-aliased rendering is integrating the radiance reaching a pixel by using a pixel-width cone-tracing through the scene. For representations allowing LoD pre-integration (like texture MIPmap and several SVO approaches mentioned above), setting the LoD according to the pixel footprint is a differential approximation of the cone integration. We revisit SVO rendering integration following this formalism.

Our hierarchical representation traversed by the cone contains view-dependent pre-integrated geometry. We consider a cone as a set of successive cone elements locally similar to cylinders, whose length equals their diameter. These cone elements constitute the neighborhood over which we integrate the microscopic rendering. Cone tracing ensures the macroscopic integration. Shadow-ray cones are treated similarly, launched from contributing cone elements, and of a size such as to fit to cone element and to light source diameters. By pre-computing a hierarchy of neighborhoods, we use the local cone diameter to access the correct MIPmap level. Thereby, our rendering scheme is similar to volume rendering with differential cones [CNLE09], but our storage and shading of voxels account for subpixel occlusions and correlation effects. This model ensures a rendering with nearly constant computational complexity. It provides smooth transitions between scales, by progressively merging the macrogeometry into the microgeometry as the MIPmap level increases. We thus get an anti-aliased and coherent rendering at the different scales that reproduces view-dependent macro- and microgeometric effects.

4.1 Cone Tracing

In a perspective camera model, a pixel value is the light intensity I perceived over a solid angle Ω. To each direction ω with solid angle $\mathrm{d}\omega$ corresponds a ray leaving the pixel (this generalizes easily to cameras with lenses and depth-of-field). The intersection of the geometry A at distance z for the ray going through ω is a binary value $\mathbb{1}_A(w,z) \in \{0,1\}$ and its visibility is given by $1-\mathbb{1}_A(w,[0,z[)$. The visible occlusion distribution produced by the geometry

$$\alpha(\omega,z) = \mathbb{1}_A(\omega,z)\ (1-\mathbb{1}_A(\omega,[0,z[)) \tag{7}$$

expresses the fact that the ray going through ω is occluded by the geometry exactly at distance z. The light intensity I perceived at the pixel, and reflected by the geometry of the scene, is the sum of the visible outgoing radiances L towards the viewer's direction. A localized description of this integral expresses it as the sum of the accumulated radiances through the space covered by the cone

$$I = \int_0^{\infty} \int_{\Omega} \alpha(\omega,z) L(\omega,z)\ \mathrm{d}\omega \mathrm{d}z \tag{8}$$

where, for each cone section at distance z along the ray, point (ω,z) is the intersection of the cone section with the ray associated to the direction ω. $L(\omega,z)$ is the outgoing radiance of the coincident surface $\{\ (\omega,z) \mid \mathbb{1}(\omega,z)=1\}$ at the intersection of the visible geometry and the cone section (Figure 3).

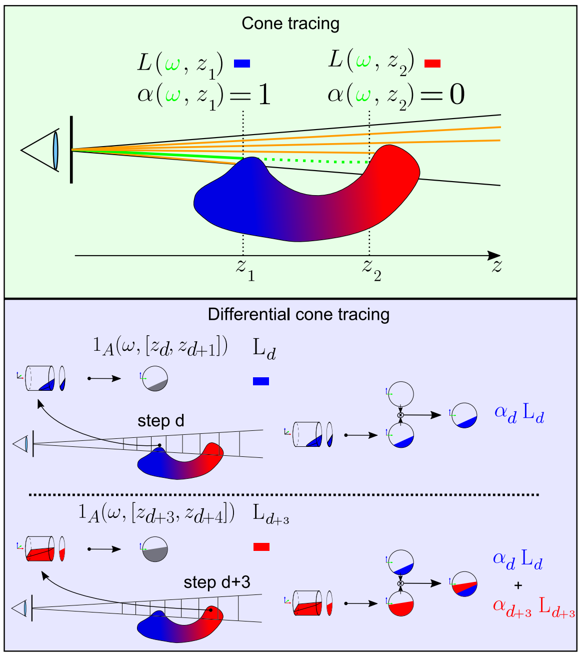

Figure 3: Comparison between cone tracing and differential cone tracing.

4.2 Differential Cone Tracing

In the perspective of local neighborhood pre-integration to ensure efficiency and scalability, we split the cone in successive *cone elements*

$$I = \sum_{d=0}^{\infty} \int_{z_d}^{z_{d+1}} \int_{\Omega} \alpha(\omega,z) L(\omega,z)\ \mathrm{d}\omega \mathrm{d}z \tag{9}$$

To permit pre-filtering, our objective is to find a way to represent the pre-integrated local visible occlusion $\alpha_d = \int_{z_d}^{z_{d+1}} \int_{\Omega} \alpha(\omega,z)\ \mathrm{d}\omega \mathrm{d}z$ and the mean local visible outgoing radiance $L_d = \frac{\int_{z_d}^{z_{d+1}} \int_{\Omega} \alpha(\omega,z) L(\omega,z)\ \mathrm{d}\omega \mathrm{d}z}{\int_{z_d}^{z_{d+1}} \int_{\Omega} \alpha(\omega,z)\ \mathrm{d}\omega \mathrm{d}z}$ in a cone element. Then, at runtime we only need to compute $I = \sum_{d=0}^{\infty} \alpha_d L_d$ (Figure 3).

L_d and α_d represent a pre-filtered element. They are not scalars, but anisotropic view-dependent functions. The two next subsections explain how to compute them.

4.3 Visible Occlusion Distribution α_d in a Cone Element

The visible occlusion in the d^{th} cone element

$$\alpha_d = \int_{z_d}^{z_{d+1}} \int_{\Omega} \mathbb{1}_A(\omega,z)\ (1-\mathbb{1}_A(\omega,[0,z[))\ \mathrm{d}\omega \mathrm{d}z \tag{10}$$

can be rewritten as

$$\alpha_d = \int_{\Omega} \mathbb{1}_A(\omega,[z_d,z_{d+1}])\ (1-\mathbb{1}_A(\omega,[0,z_d[)\ \mathrm{d}\omega \tag{11}$$

where $\mathbb{1}_A(\omega,[z_d,z_{d+1}])$ is the indicator function of the intersection of the ray going through $\mathbf{x}$ and surface A in

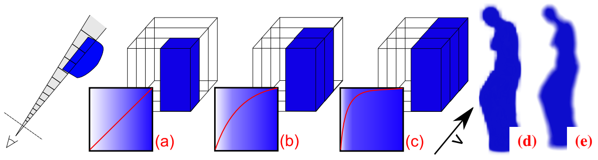

Figure 4: (a): *A pixel is only half covered by opaque geometry, thus the fragment has an opacity* $\alpha = 0.5$. (b,c): *Successively accumulating opacity by naive blending progressively saturates the final result, while it should stick to 0.5, i.e., the* α*-blending model is wrong when fragments (or successive cone elements) are highly correlated, which is the case at silhouettes. This tends to thicken silhouettes and become especially visible with depth-of-field* (d). *Our model takes into account correlations along the ray and produces correct silhouettes even with depth-of-field* (e).

the cone element $[z_d, z_{d+1}]$. The product of the indicator functions in the integral expresses the correlation between the intersections events along different rays. If we suppose them uncorrelated, we could integrate them in a separable way

$$\alpha_d = \int_\Omega \mathbb{1}_A(\omega,[z_d,z_{d+1}])\ \mathrm{d}\omega \int_\Omega (1-\mathbb{1}_A(\omega,[0,z_d[)\ \mathrm{d}\omega \tag{12}$$

which corresponds to the blending model $\alpha_d = \alpha_{[z_d,z_{d+1}]}(1-\alpha_{d-1})$ used in volume rendering [KVH84]. Indeed, in volume rendering the opacity α of a voxel represents the occlusions produced by an important amount of microscopic elements statistically uncorrelated along a ray. Yet, this uncorrelation hypothesis is not valid in the case of occlusions produced by dense objects with well-contrasted spatial distributions. Neglecting this produces errors such as excessive opacity accumulation along silhouettes (see Figure 4). A good rendering model should thus take the correlation between the terms in integral (11) into account. Evaluating α_d requires to represent and to manipulate the distributions $\mathbb{1}_A(\omega,[z_d,z_{d+1}])$.

4.4 Outgoing Radiance L_d in a Cone Element

We make the hypothesis **(H$_1$)** that correlation between radiance and visibility only exists at the neighborhood's scale, and that there is no correlation between faraway occlusion and local radiance. This allows us to consider local L_i independently. We have $L_d \approx \frac{\int_{z_d}^{z_{d+1}}\int_\Omega \mathbb{1}_A(\omega,z)\, L(\omega,z)\ \mathrm{d}\omega \mathrm{d}z}{\int_{z_d}^{z_{d+1}}\int_\Omega \mathbb{1}_A(\omega,z)\ \mathrm{d}\omega \mathrm{d}z}$ by canceling out the term $1-\mathbb{1}_A(\omega,z)$ that gets out of the integral thanks to uncorrelation hypothesis.

To compute L_d, we need a model that describes the geometry inside the d^{th} cone element, a model for the distribution of the surface attributes (we propose one in Section 5), and the analytical integration of the masking and shadowing effects on these attributes over a complex surface (Figure 3). By considering the correlation between the surface attributes and their visibility, we get a similar form of Eq. (3)

$$L \approx \bar{E}_\mathbf{l}\ \bar{a}(\mathbf{v},\mathbf{l})\ \bar{\rho}(\mathbf{v},\mathbf{l})\ \mathbf{l}\bar{\mathbf{n}}\ \bar{V}(\mathbf{l}) \tag{13}$$

in which we replace $\bar{a}$ by the mean visible attribute $\bar{a}(\mathbf{v},\mathbf{l})$ given by

$$\bar{a}(\mathbf{v},\mathbf{l}) = \frac{\int_A a(\mathbf{x})\ V(\mathbf{v},\mathbf{x})V(\mathbf{l},\mathbf{x})\ \mathbf{vn}(\mathbf{x})\ \mathrm{d}\mathbf{x}}{\int_A V(\mathbf{v},\mathbf{x})V(\mathbf{l},\mathbf{x})\ \mathbf{vn}(\mathbf{x})\ \mathrm{d}\mathbf{x}} \tag{14}$$

From Eq. (3), we only keep the earlier hypothesis of far-away uncorrelation between radiance and occlusion which allows to take $E_\mathbf{l}$ out of Integral (2) (this is already in **(H$_1$)**), and the hypothesis of uncorrelation between $a_i(\mathbf{x})$ and $\rho_i(\mathbf{x},\mathbf{n}(\mathbf{x}))$, let us denote it **(H$_2$)**.

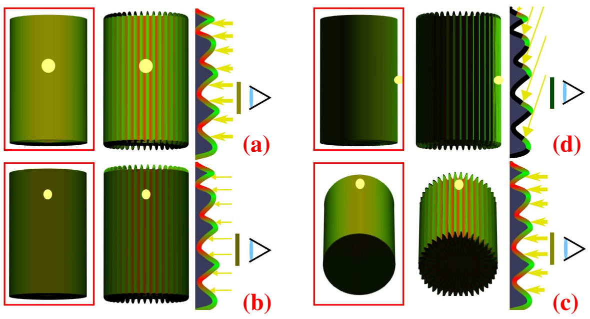

Figure 5: Correlation between the surface attributes and their visibility. *The red boxed images show how our shading model (Eq. (13)) reproduces geometric occlusion effects on an anisotropic surface.* (a) *View and light are normal to the surface, there are no light- or view-dependent effects.* (b,c) *The light and the camera are moved to a grazing angle parallel to the direction in which the anisotropic surface is constant. Still no view-dependent effects.* (d) *The light is moved to a grazing angle parallel to the direction in which the anisotropic surface oscillates. The green bumps stay in the lighted zone while the red grooves disappear in the shadow making dark green the resulting average color.*

5 Our Computation Model

In this section, we propose a way to represent the microgeometry and the attributes distributions in order to calculate Eqs. (11) and (14).

5.1 Hypotheses

We base our approach on five additional hypotheses :

(H$_3$) The microgeometry is represented with a Gaussian surface [Smi67], possibly anisotropic. This common choice is justified by the compactness of such a representation, the simplicity of computing, interpolating, and manipulating its parameters, as well as the properties that can be analytically derived from it.

(H$_4$) BRDF $\rho_i(\mathbf{x},\mathbf{n}(\mathbf{x}))$ and depth h of the surface are uncorrelated, in particular normals with respect to depth (for applicability of [Smi67]. But it is already a consequence of **(H$_3$)**).

(H$_5$) We assume surface attributes a_i are correlated only with their depths h within the surface (which is the case for many real surfaces). This allows to separate $\int a(\mathbf{x})$ and $\int \mathbf{n}(\mathbf{x})$ in Eq. (14).

(H$_{5bis}$) We assume that the distributions of the average attributes values can be represented as a function of the heights of the surface details: $a(h) = \bar{a} + a_s s(h)$, where $\bar{a}$ is the mean value of the attribute and $s(h)$ a centered and normalized increasing function. It is interesting to take a sigmoid function s to avoid spoiling the dynamics of a in loosely representative extrema. We choose $s(h) = 2g(h) - 1$ with $g(h)$ from Eq. (5) which enables the analytical integration of Eq. (14). Parameter a_s represents the correlation between h and a.

(H$_6$) The macrosurface is locally planar, i.e.the macroscopic curvature does not interfere with the computation of the Gaussian parameters, like in most of the previous work on surface attributes pre-filtering [BN11]. Our computation of visibility V is also based on that approximation. This hypothesis fixes the validity domain of our model.

(H$_7$) The macrosurface belongs to a B-rep object. We do not represent thin objects whose inside parts cannot be captured by the resolution of the voxels.

5.2 Voxel-based Data Structure Representation

We use the octree structure from Crassin et al. [CNLE09], with n^3 voxel bricks stored at each node, which makes hardware interpolation between voxels possible. We consider volumetric objects as macroscopic signed distance fields with statistical descriptors of the microscopic behavior. Figure 6 shows the data we store in each voxel

- Macroscopic distance field $\bar{h}$ and the variance of its microscopic oscillation amplitudes σ_h^2
- Macroscopic normal $\bar{\mathbf{n}}$ and the roughness σ_n^2 of the microgeometry. The associated NDF is a Gaussian lobe with mean $\bar{\mathbf{n}}$ and slope variance σ_n^2 ($\sigma_{n_x}^2$ and $\sigma_{n_y}^2$ in the anisotropic case)
- Microscopic distributions of each attribute with $\bar{a}$ and a_s. Representing RGB colors thus requires $3\times$ these two parameters.

Note that we store and interpolate variances σ^2, which is the quantity that interpolates linearly (and is thus suited for hardware interpolation).

Pre-computation We pre-compute a hierarchical representation of filtered attributes and geometric details. Each parameter p described above is initialized at the deepest level from the corresponding input data as $\bar{p} = p$ and $\sigma_p^2 = 0$. The statistics of the parameter at each scale are computed as an integral in the deepest level over the corresponding neighborhood. We compute $\bar{h}$, $\bar{\mathbf{n}}$, and $\bar{a}$ (mean values) and σ_h^2 and σ_n^2 (variances) with the usual statistic formulas. We compute projection $a_s = \frac{\int s(h(x))a(x)\,\mathrm{d}x}{\int s^2(h(x))\,\mathrm{d}x}$ of attribute a on the function $s(h)$ **(H$_{5bis}$)** parametrized by σ_h (see Eq. (5)).

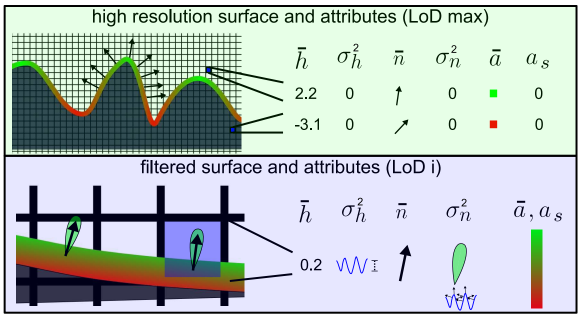
high resolution surface and attributes (LoD max)
$\bar{h}$ σ_h^2 $\bar{n}$ σ_n^2 $\bar{a}$ a_s
2.2 0 0 0
-3.1 0 0 0
filtered surface and attributes (LoD i)
$\bar{h}$ σ_h^2 $\bar{n}$ σ_n^2 $\bar{a}, a_s$
0.2

Figure 6: Our voxel representation of surfaces with attributes.

Memory Footprint We use two channels for the distance field parameters ($\bar{h}$ and σ_h^2), four (isotropic) or five (anisotropic) channels for the macro-normal ($\bar{\mathbf{n}}$) and the micro-NDF (σ_n^2) and two channels per surface attribute ($\bar{a}$ and a_s), e.g., color channels. While 32-bit precision is preferable for the distance field, 8- or 16-bit channels are reasonable for the other components. Thus, our representation handles multi-scale geometry, view-dependent filtered RGB colors and shading for an average 15-20 bytes per voxel (possibly less at the deepest level where $\sigma_p^2 = 0$). This is about two or three times as much as in [CNLE09] with RGBA values and normals.

5.3 Overall Algorithm

We use the octree traversal algorithm described in [CNLE09] to sample the voxel field. Algorithm 1 explains how we use the voxel data structure to achieve practical calculations of illumination and occlusion at runtime and is illustrated in Figure 7.

Algorithm 1 Cone tracing for one pixel

```
1:  $d = 0$ : cone element index
2:  vec3 $\mathbf{p}(d)$ : cone element $d$ center's position
3:  int $\alpha = 0$ : binary mask with $N$ bits
4:  float $L = 0$ : mean pixel incoming radiance
5:  while $\mathbf{p}(d)$ in volume data do
6:      compute cone width $w_d$ and MIPmap level
7:      sample voxel data at $\mathbf{p}(d)$ at the proper MIPmap level
        get $\bar{h}$, $\sigma_h^2$, $\bar{\mathbf{n}}$, $\sigma_n^2$, $\bar{a}$, $a_s$
8:      compute float $\theta(\bar{h}, \bar{\mathbf{n}})$ and float $v(\bar{h}, \bar{\mathbf{n}})$ (5.4)
        get int $\mathbb{1}_A(\theta, v)$ (texture fetch)
9:      compute float $L_d(\bar{n}, \sigma_n^2, \sigma_h^2, \bar{a}, a_s)$ (5.5)
10:     $L = L + L_d \frac{\text{bitcount}(\mathbb{1}_A \setminus \alpha)}{N}$
11:     $\alpha = \alpha \cup \mathbb{1}_A$
12:     $d = d + 1$
13: end while
14: return $L$
```

5.4 Computation of the Occlusion Distribution

This section proposes a representation and an algorithm to compute $\mathbb{1}_A(\omega,[z_d,z_{d+1}])$ necessary for the evaluation of Eq. (11). According to hypothesis **(H$_6$)**, the mean geometry can be locally represented by the plane specified by the signed distance $\bar{h}$ and the normal $\bar{n}=(n_x,n_y,n_z)$. This plane defines a half-space whose 3D intersection with the cone element gives a 2D occupancy distribution over the pixel footprint which is computed analytically. It enables us to compute the contribution of the local geometry to the pixel, while taking into account the correlations with occlusions along the cone's axis. We associate a tabulated mask to that distribution to represent the functions $\mathbb{1}_A$. We can thus compute and combine them efficiently as in [Car84].

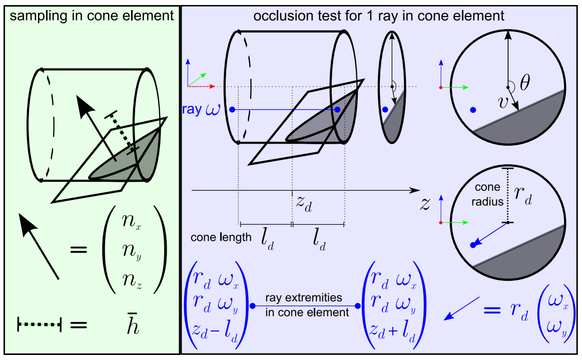
sampling in cone element
$= \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$
$= \bar{h}$
occlusion test for 1 ray in cone element
ray ω
θ
v
z_d
z
cone length l_d l_d
cone radius r_d
$\begin{pmatrix} r_d\,\omega_x \\ r_d\,\omega_y \\ z_d-l_d \end{pmatrix}$
ray extremities in cone element
$\begin{pmatrix} r_d\,\omega_x \\ r_d\,\omega_y \\ z_d+l_d \end{pmatrix}$
$= r_d \begin{pmatrix} \omega_x \\ \omega_y \end{pmatrix}$

Figure 7: Computation of the mask in a cone element. Left: *The data are sampled at the cone center. The equation of the plane tangent to the geometry is given by* $\bar{h}$ *and* $\bar{\mathbf{n}}$. Right: *We test for a given ray if it lays in the part of the pixel footprint covered by the geometry.*

Computation of the masks The cone element d is locally approximated with a cylinder at distance z_d from the eye and oriented in direction $\vec{\mathbf{z}}$, of radius r_d, and length l_d. The binary mask is a set of points (ω_x,ω_y) on the pixel footprint associated with the rays going through the directions ω. For each ray (ω_x,ω_y) we compute if it intersects the plane with normal (n_x,n_y,n_z) and distance $\bar{h}$ (see Figure 7). This plane is given by the equation $xn_x+yn_y+(z-z_d)n_z+\bar{h}=0$. The ray passing through ω intersects the geometry in the cone element if at least one of the extremities $(r_d\omega_x, r_d\omega_y, z_d \pm l_d)$ is below the plane: $r_d\omega_x n_x + r_d\omega_y n_y - l_d n_z + \bar{h} \leq 0$. The intersection test for the bit of the mask associated with point (ω_x,ω_y) of the pixel is then $\omega_x n_x + \omega_y n_y \leq \frac{n_z l_d - \bar{h}}{r_d}$. We rewrite the projection of the normal on the pixel footprint $(n_x,n_y)=\sqrt{n_x^2+n_y^2}(\cos\theta,\sin\theta)$ in polar coordinates and the final intersection test has the form $\omega_x\cos\theta+\omega_y\sin\theta \leq v$ with $v=\frac{n_z l_d-\bar{h}}{r_d\sqrt{n_x^2+n_y^2}}$ (see Figure 7). The state of each bit (ω_x,ω_y) of the mask and thus the distribution $\mathbb{1}_A(\omega,[z_d,z_{d+1}])$ is then entirely described with the two parameters (θ,v). We pre-compute each mask and store it as an integer value in a 2D texture parametrized by (θ,v). At the runtime, for each cone element d, we compute θ and v and fetch the texture in nearest mode to get the mask.

5.5 Computation of the Local Illumination

When the cone intersects the geometry, the radiance emitted by the geometry contributes to a part of the pixel. This section focuses on the representation and on the computation of the BRDF of the microscopic surface and of the view-dependent mean surface attributes $\bar{a}(\mathbf{v},\mathbf{l})$ involved in the computation of the outgoing radiance (Eq. (13)).

BRDF Representation To simplify the convolution of NDFs with BRDFs, we assume as in previous work that both can be represented in the same way. We rely on their Gaussian slope statistics $\mathcal{N}(\bar{\mathbf{n}},\sigma_n^2)$ representation [CT81, ON94]. The initial microfacet statistics of the BRDF $\sigma_{n_\rho}^2$ is progressively enriched with the filtering of meso-surface normals σ_n^2. Convolving two random Gaussian variables comes down to adding the variances. At runtime, we compute the shading with the convolved BRDF with variance $\sigma_n^2+\sigma_{n_\rho}^2$.

View-dependent attributes According to **(H$_5$)**, attribute $a(h)$ and visibilities $V(\mathbf{v},h)$ (from the eye) and $V(\mathbf{l},h)$ (from the light source) are expressed as functions of h. We reformulate Eq. (14) by integrating over h:

$$\bar{a}(\mathbf{v},\mathbf{l}) = \frac{\int_{-\infty}^{\infty} a(h)\,V(\mathbf{v},h)V(\mathbf{l},h)\,P(h)\,\mathrm{d}h}{\int_{-\infty}^{\infty} V(\mathbf{v},h)V(\mathbf{l},h)\,P(h)\,\mathrm{d}h} \tag{15}$$

where the microscopic surface has heights with distribution $\mathcal{N}(0,\sigma_{\bar{h}}^2)$, attributes $a(h)=\bar{a}+a_s s(h)$, and visibility probability $V(\mathbf{d},h)$ given by Smith's model (Eq. (4)) for direction $\mathbf{d}$.

In Eq. (15), we can expand $a(h)$ out of the integral. According to Smith's model, we have $P(h)=g'(h)$ and $V(\mathbf{v},h)=g(h)^{\Lambda(\mathbf{v})}$. Eq. (15) hence becomes

$$\bar{a}(\mathbf{v},\mathbf{l}) = \bar{a} - a_s + 2a_s \frac{\displaystyle\int_{-\infty}^{\infty} g'(h)\,g(h)^{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})+1}\,\mathrm{d}h}{\displaystyle\int_{-\infty}^{\infty} g'(h)\,g(h)^{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})}\,\mathrm{d}h} \tag{16}$$

which has the following analytical solution

$$\bar{a}(\mathbf{v},\mathbf{l}) = \bar{a} + a_s\left(2\frac{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})+1}{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})+2} - 1\right) \tag{17}$$

6 Implementation and Results

We implemented our algorithm in CUDA on an NVIDIA GTX 560 graphics card in a PC with an Intel Core 2.40 GHz and 8 GB memory. Our SVO implementation (data management, octree traversal and sampling) is essentially based on the voxel engine presented in [CNLE09] in which we added the stages described in Sections 5.4 and 5.5.

In the following results, our images are rendered with a resolution of 512×512. The typical performances are 40-60 fps without shadows and 10-25 fps with shadows (in the following, if not explicitly mentioned, performances are without shadows). While zooming in, the cost per covered pixel is nearly constant around 0.1-0.3 μs/pixel. This cost mainly depends on the presence of silhouettes: views with no silhouettes are the fastest, views with large grazing areas are the most expensive since several cone elements per ray are computed.

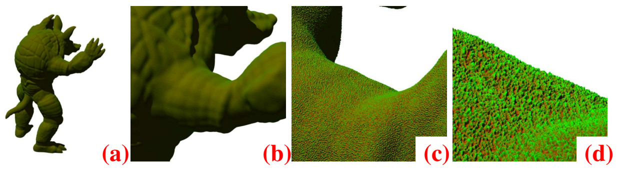

Figure 8: *Our model shows seamless transitions when zooming in (see also the video).*

View Fig. 8	far view 8.a	mid view 8.b	close view 8.c	closer (silh) 8.d	closer (no silh)	far DoF	far shadow
fps	57	37	25	19	66	110	26
ms	17.5	27	40	52.6	15	9	38.5
μs/pix	.26	.13	.22	.32	.06	.13	.57

The importance of interpolation for good-looking SVOs is illustrated in Figure 10. Carmack's SVO are blocky and aliased due to the lack of interpolation. Crassin et al. [CNLE09] reveals cubical patterns since opacity is not a correct descriptor for occlusion correlated along a ray. ESVOs (not figured here) encode a subvoxel 3D boundary yielding a sharp polygonal-like magnification. But it is aliased and looks polyhedral by lack of integration and interpolation, and it suffers from parallax shifting under animation due to the *nearest* operator. Our method ensures anti-aliased sharp magnification as well as temporal and zooming in and out coherency. The three methods compared here achieve the same performances. This means that the computational overhead introduced by our algorithm is negligible in comparison to the time spent in the other parts of the algorithm (data management, octree traversal and sampling). The dataset is a voxelized Marko Dabrovic's Sponza. The octree has a resolution of 2048^3 voxels and occupies 8 GB. We use masks (see Section 5.4) with 128 Poisson-distributed samples, so that atomic mask operations are done using four 32-bit integers. Our pre-calculed mask table is 256×256.

Anti-aliasing Our method ensures proper anti-aliasing of silhouettes even for complex subgeometry and correlated fragments, at very good performances, when classical solutions are either costly (oversampling) or biased (see Figure 4(d) for cone rendering on volume densities). Indeed, our scheme works exactly the same for depth-of-fields, yielding even better performances: As for [CNLE09], our cone-tracing scheme is faster for depth-of-fields (see Figure 1(d)) than for focussed images, as the former relies on coarser LoDs.

Material Filtering We demonstrate how our method is able to filter correctly view-dependency on real material (Figure 9). The plots compare the groundtruth and separate color MIPmapping with the output of our model (Eq. (17)). Color variation becomes an important feature at grazing angles (especially at silhouettes like in Figure 11) and are well captured by our model while seperate color MIPmapping is not view-dependent at all. The effects of surface anisotropy ($\sigma^2_{n_x} \neq \sigma^2_{n_y}$) and combined light- and view-dependency are illustrated on the cylinders in Figure 5. Our model can also be used directly as a material editor without the burden of managing explicit details (see Figure 1(e)). In such case the shader has to evaluate Eq. (17) which is possible with a few lines of code and easy to insert in an existing rendering pipeline.

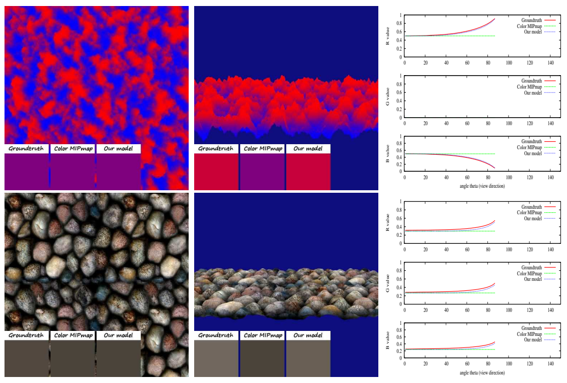

Figure 9: Testing our method using materials having height/color correlation. Left: *Comparison for two typical view angles of our resulting pixel color against the groundtruth (obtained by averaging a high-res image) and MIPmapping for view-dependency only (lighting is disabled and the term $\Lambda(\mathbf{l})$ is set to 0 in Eq. (17)).* Right: *detailed comparison of R,G,B curves for varying view angles. Our model is view-dependent and fits pretty well the groundtruth (especially for real-time usage) while MIPmapping is constant and correct only for normal view angles.* Top: *The maximum error of our method is less than 1% on a Perlin noise height map (which tends to produce a Gaussian surface ($\boldsymbol{H_3}$) and with a color $s(h)$ of the height ($\boldsymbol{H_5}$) and ($\boldsymbol{H_{5bis}}$).* Bottom: *On a real-world texture which is not really Gaussian, the error is about 5% at grazing angles.*

Accurate 3D Filtering Our algorithm is able to correctly reproduce subpixel color effects due to correlated visibility from the eye and from the light source, that are comparable to the groundtruth, contrary to the naive method processing separate filtering of geometry and color (see Figure 11). In particular, we ensure seamless zoom with close to no color shift (see Figure 8) and correct transformation of meso-granularity to BRDF roughness (see Figure 12) while keeping good real-time performances. See also the companion video. The voxel octree containing the data in

Figures 11 and 12 has a resolution of 512^3 and requires 300 MB storage on the GPU. To obtain enough data for really deep zoom, we further enhance these details with 3 to 8 octaves of 3D Perlin noise: The close views have a virtual resolution of 8192^3. Note that our representation (based on a distance field) allows procedural surface enhancement which is not possible with contour data [LK10a] or blurry opacity [CNLE09].

7 Conclusion and Future Work

In this paper, we have presented a new multiscale surface representation and a rendering algorithm able to reproduce view-dependent effects of detailed geometry accounting for correlation of occlusion and attributes with visibility. We have shown how our algorithm handles deep zoom, and maintains coherency through scales while achieving real-time results. We produce anti-aliased constant-cost accurate effects in real-time, making the management of very detailed objects scalable without compromising quality or performances.

Our contributions are two folds: a theoretical framework, and a computational model with stronger practical hypotheses. We described explicitly our hypotheses and limitations along the paper. Indeed, we consider our model as a step toward the real-time rendering of complex geometry with smooth and coherent transitions between many scales. Here, we released as much as possible non-valid or restrictive hypothesis of common pre-filtering schemes. Among the limitations of our current representation, the macro B-rep assumption could probably be released through the management of thin parts, like the 2-sided sheets as in [CNS*11]. The management of reflection and refraction as secondary differential cone is already described in [Ige99]. Beside the extra cost, the complication mainly stands in the current lack of recursive *threads* in Cuda. At least, reflection toward environment maps should be an tractable extension. In the scope of animation one could use pseudo-volumetric structures such as shellmaps.

Still, more complex configurations exist, and deep filtering remains a "Holy Grail", starting with accounting for the curvature of coarse surfaces. This leaves a lot of interesting problems to solve. For instance, really complex surfaces or subpixel details no longer behave like surfaces, but like volumes at a distance (grass, wire mesh, foliage, semi-transparent material, etc.). Our volume implementation assumed opaque objects with defined coarse surfaces. Adapting it to the filtering of view-dependent effects in semi-transparent volumes would be another interesting but challenging future work.

Acknowledgements
We thank Charles de Rousiers for his precious advices and and Pierre Poulin for his careful proofreading.

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). 2, 3, 6

[Car84] Carpenter L.: The A-buffer, an antialiased hidden surface method. In *Proceedings of SIGGRAPH '84* (1984), pp. 103–108. 3, 7

[Car08] Carmack J.: John Carmack on id Tech 6. Interview in PC Perspective, 2008. 1, 2, 10

[CNLE09] Crassin C., Neyret F., Lefebvre S., Eisemann E.: Gigavoxels : Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of I3D '09* (2009). 2, 3, 4, 6, 7, 8, 9, 10

[CNS*11] Crassin C., Neyret F., Sainz M., Green S., Eisemann E.: Interactive indirect illumination using voxel cone tracing. *Proceedings of Pacific Graphics 2011 30*, 7 (2011). 2, 9

[CT81] Cook R. L., Torrance K. E.: A reflectance model for computer graphics. In *Proceedings of SIGGRAPH '81* (1981), pp. 307–316. 2, 3, 7

[Fou92] Fournier A.: Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination* (1992), pp. 45–52. 2

[HSRG07] Han C., Sun B., Ramamoorthi R., Grinspun E.: Frequency domain normal map filtering. In *Proceedings of SIGGRAPH '07* (2007). 2

[Ige99] Igehy H.: Tracing ray differentials. In *Proceedings of SIGGRAPH '99* (1999), ACM, pp. 179–186. 2, 9

[KVH84] Kajiya J. T., Von Herzen B. P.: Ray tracing volume densities. In *Proceedings of SIGGRAPH '84* (1984), pp. 165–174. 5

[LFTG97] Lafortune E. P. F., Foo S.-C., Torrance K. E., Greenberg D. P.: Non-linear approximation of reflectance functions. In *Proceedings of SIGGRAPH '97* (1997), pp. 117–126. 3

[LK10a] Laine S., Karras T.: Efficient sparse voxel octrees. In *Proceedings of ACM SIGGRAPH 2010 Symposium on Interactive 3D Graphics and Games* (2010), pp. 55–63. 2, 9

[LK10b] Laine S., Karras T.: Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics 17* (2010), 1048–1059. 2

[LK10c] Laine S., Karras T.: *Efficient Sparse Voxel Octrees – Analysis, Extensions, and Implementation*. NVIDIA Technical Report NVR-2010-001, NVIDIA Corporation, Feb. 2010. 2

[OB10] Olano M., Baker D.: Lean mapping. In *Proceedings of ACM SIGGRAPH 2010 Symposium on Interactive 3D Graphics and Games* (2010), I3D '10, ACM, pp. 181–188. 2

[ON94] Oren M., Nayar S. K.: Generalization of Lambert's reflectance model. In *Proceedings of SIGGRAPH '94* (1994), pp. 239–246. 2, 3, 7

[Pho75] Phong B. T.: Illumination for computer generated pictures. *ACM 18* (June 1975), 311–317. 3

[Smi67] Smith B.: Geometrical shadowing of a random rough surface. *IEEE Transactions on Antennas and Propagation 15* (1967), 668–671. 2, 3, 5, 6

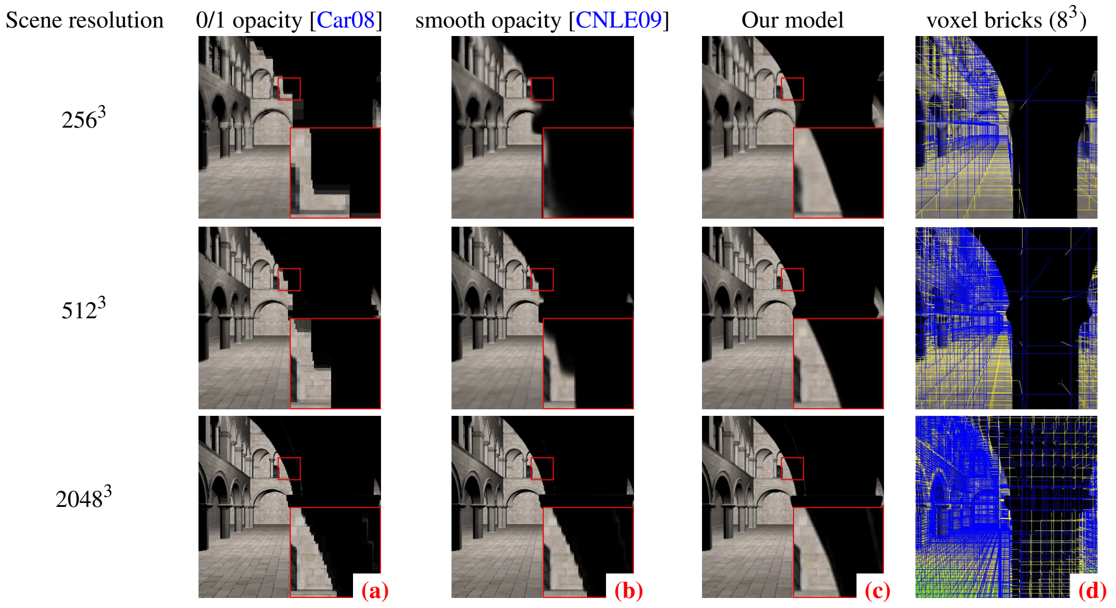

Figure 10: Comparison of SVO interpolation and magnification quality. (a) *Subvoxel geometry is represented as 0/1 opacity, nearest value is used at samples along the ray (as in [Car08]).* (b) *Using smooth opacity (α values), quadrilinearly interpolated at samples along the ray ([CNLE09] and volume rendering).* (c) *Subvoxel geometry is represented using our 3D mask, quadrilinearly interpolated at samples along the ray.* (d) *Bricks (yellow) of* 8^3 *voxels and empty nodes (blue).*

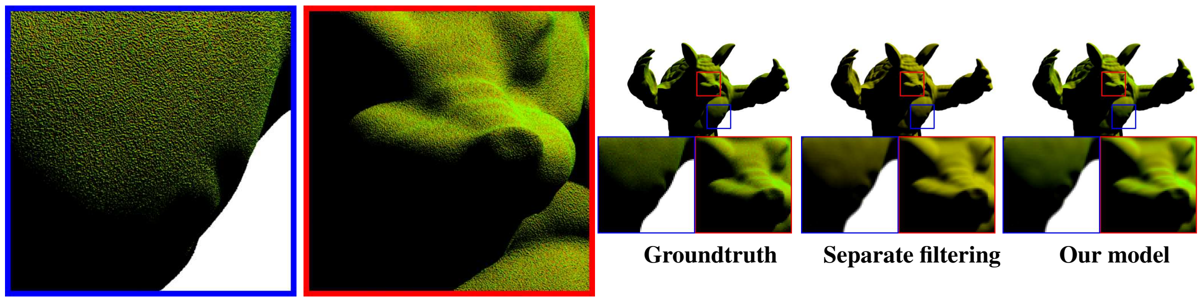

Figure 11: Comparisons of light- and view-dependent color effects. *Grazing light or view directions cancel out the contribution of colors correlated to deep locations (here, the red) as seen in the two regions of interest. Average color shifts from yellow to green. Naive separate filtering of colormap gives uniform yellow, while our model reproduces the groundtruth.*

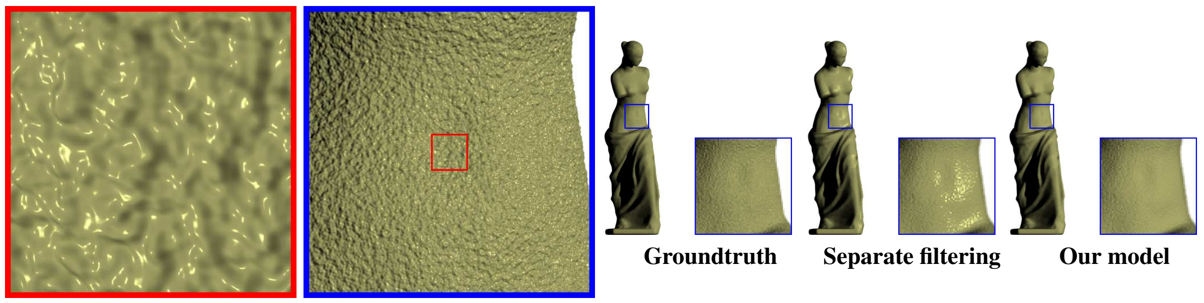

Figure 12: Comparisons of emboss-to-shading filtering. *A bumpy specular area appears diffuse at distance. With a correct filtering, details go from geometry to BRDF. Naive separate filtering of normals applied to the base BRDF gives a wrong shading, while our model reproduces the groundtruth.*