A Visibility-Aware Model for Pre-Filtering and Rendering Surfaces in Real-Time

Eric Heitz, Fabrice Neyret

A Visibility-Aware Model for Pre-Filtering and Rendering Surfaces in Real-Time

Eric Heitz and Fabrice Neyret

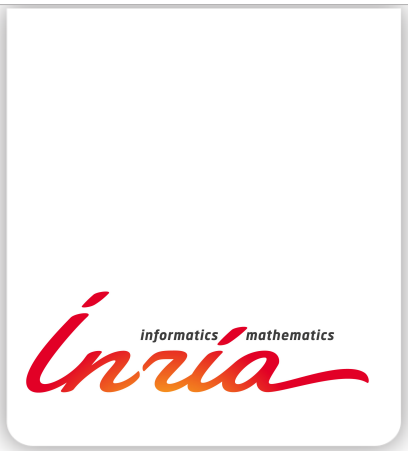
RESEARCH REPORT

N° 7827

December 2011

INRIA Grenoble Rhône-Alpes,
Université de Grenoble et CNRS,
Laboratoire Jean Kuntzmann

ISSN 0249-6399 ISRN INRIA/RR--7827--FR+ENG



A Visibility-Aware Model for Pre-Filtering and Rendering Surfaces in Real-Time

Eric Heitz and Fabrice Neyret

INRIA Grenoble Rhône-Alpes, Université de Grenoble et CNRS,
Laboratoire Jean Kuntzmann

Research Report n° 7827 — December 2011 — 23 pages

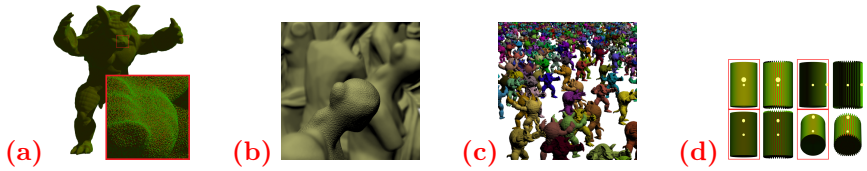


Figure 1: *Our pre-filtering representation and rendering algorithm reproduce in real-time the complex shading effects due to subpixel details. These are due to various correlation effects usually neglected in real-time methods like MIPmapping (cf comparisons in Figure 9 and 8). This allows for correct color varying effects* (a,d)*, emboss-to-shading filtering* (b)*, anti-aliasing* (c) *and depth of field* (b) *without oversampling, with seamless transitions at zooming or defocusing* (a,b).

Abstract: We present a multiscale surface appearance representation and a rendering model that accounts for the subpixel visibility distribution. Starting from this model, we propose a method for pre-filtering detailed surfaces and their attributes. Our representation of the filtered attributes takes the correlation with their visibility into account. The masking and shadowing effects lost in geometric filtering of the surface can thus be recovered at rendering. This grants high visual quality of subpixel effects while ensuring a nearly constant per pixel computation complexity. Besides, accounting for subpixel occlusion naturally anti-aliases geometry. The computational model we propose has low memory and computational time requirements, and is thus well-suited for real-time rendering.

Key-words: Computer Graphics - Picture/Image Generation - Three-Dimensional Graphics and Realism

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Un modèle tenant compte de la visibilité pour préfiltrer et rendre des surfaces en temps-réel

Résumé : Nous proposons une représentation et un modèle de rendu prenant en compte la distribution de visibilité sous-pixel. En partant de ce modèle, nous proposons une méthode pour préfiltrer les surfaces détaillées et leurs attributs. Notre représentation filtrée des attributs surfaciques tient compte de leur corrélation avec leur visibilité et permet de reconstruire, au moment du rendu, les effets de masquage et d'ombrage perdus dans le filtrage géométrique de la surface. Ceci permet une grande qualité visuelle des effets sous-pixel avec un seul échantillon par pixel. En outre, la prise en compte des occlusions sous-pixel antialiase naturellement la géométrie. Le modèle de calcul et l'implémentation proposés sont légers en terme de stockage mémoire et en temps de calcul et satisfont les exigences d'un rendu temps-réel.

Mots-clés : Informatique Graphique - Synthèse d'Images

1 Introduction

1.1 Motivation

Geometric models get more and more complex. The management of details is of high importance to simultaneously ensure visual quality and acceptable computation time, especially for real-time applications. A pixel's color is the sum of the contributions of all details that project into it. Each detail combines matter and texture effects, masking, and illumination. The theoretical ground truth image is obtained by sampling every detail in the scene. But when thousands of elements contribute to each pixel, sampling get massive and yields a huge computational time. In the field of real-time rendering, important oversampling is not affordable, and alternative solutions requiring ideally one single computation per pixel are needed. Still, geometric simplification methods [GH97, Hop97] do not allow correct appearance filtering: microscopic details cannot be simply removed, as they produce visually important photometric effects. The problem is thus how to filter appearance, and not simply geometry. Kajiya [Kaj85] proposes a hierarchy of models adapted to the different scales: geometry, texture, and BRDF – the BRDF being considered as a filtered version of the microgeometry. Approaches which separate the normals from the geometry [COM98] and smooth transitions from one scale to another [BM93] share this spirit. Filterable representations are necessary for the pre-computation of each LOD within a range of scales. Uncorrelation[1] and separability[2] hypothesis used in usual pre-filtering techniques (*e.g.,* colormap filtering) are invalid most of the time. Our work concentrates on accounting for such correlations in the pre-filtering process.

1.2 Contributions

In this article we propose :

• A new scheme for representing attributed surfaces, and the associated rendering process: We efficiently integrate subpixel effects such as correlation between surface attributes (*e.g.,* color, BRDF) and their visibility, or the rendering of anti-aliased complex silhouettes. Our scheme adapts the concept of a volumetric integration along a conic ray [CNLE09] by using view-dependent and pre-filtered local representations of occlusion and attributes for each cone element.

• A computational model derived of this scheme: Our computations are compatible with real-time rendering, and are applied to surfaces with (possibly anisotropic) Gaussian statistics, and to attributes correlated by their depth within the surface (a common case of correlation with visibility, as for terrain, cloth material, crackled rusty or embossed material, etc). We derive view-dependent and pre-filtered models for both occlusion and appearance. The correlation of occlusion along the cone is processed with a variant of Carpenter's subpixel masks [Car84]. Our masks are computed at runtime from our filtered 3D surface representation after adjustment with a view-dependent term.

Properties, performances and comparisons to ground truth and to classical models neglecting these correlation effects are shown in section 5. To the best of our knowledge, this is the first model assuring proper subpixel filtering of

[1] a and b are uncorrelated if $\int ab = \int a \int b$.

[2] f is separable if it can be written as $f = ab$.

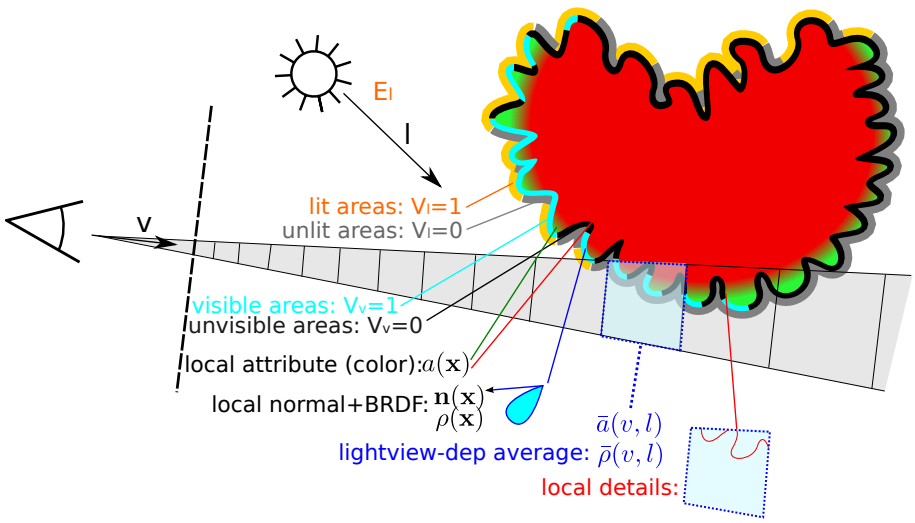


Figure 2: *Context:* rendering objects with complex subpixel details possibly correlated with depth within the coarse surface. The prefiltering must account that their contribution is correlated to their visibility from eye or light.

correlation effects and silhouettes on complex surfaces, compatible with real-time rendering pipelines as it relies on pre-filtering instead of oversampling.

2 Previous Work

We first review cone tracing and subpixel masks techniques, as our rendering model uses them to integrate microgeometry and to account for subpixel correlation. We discuss the correlation problem between surface attributes and their visibility in existing filtering techniques. Then, we dwell on the microscopic surfaces models we drew inspiration of in order to design our model.

2.1 Cone Tracing

Cone tracing [Ama84] is a variant of ray tracing in which rays are assigned a width in ways that the rendering is integrated over the entire pixel surface. The portion of the cone footprint covered by the geometry is computed analytically, and its combination with upstream masks is used to determine the contribution of each intersected surface. This method is hardly usable for complex objects. Heckbert and Hanrahan [HH84] propose beam tracing as an oversampling technique. They associate a list of intersections with polygons to each beam. Each intersection can produce reflection or refraction which are represented with new beams. Neyret and Crassin [Ney98, CNLE09] extend the notion of differential cone tracing introduced by the MIPmaps [Wil83] to the 3D case : the width of the cone determines a neighborhood size used to fetch hierarchically pre-filtered data. Data is linearly filtered without any notion of correlation or masking.

[CNLE09] introduces a directional dependency essentially limited to the separation of two faces of the same surface. Thomas et al. [TNF89] propose to surround objects with in- and outdoors proxy-surfaces, as the detection of the intersection of a blurry ray with a surface is equivalent to detecting the intersection of a ray with a blurry surface. This blurry surface idea is a first step towards the use of pre-filtered geometry such as the one we use in our model.

2.2 Correlation Between Subpixel Occlusion

A pixel's occlusion can be efficiently represented with an opacity α only if fragments are not correlated inside a pixel. This assumption is often false, especially along facets' borders or objects' silhouettes (see Figure 4). To account for such spatial correlations, A-Buffer techniques [Car84, AW85, Sch91] use binary masks for representing spatial occlusion distributions, usually encoded with subpixel bit arrays. Even though these techniques allow correct anti-aliasing of edges, they simply delegate the solving of filtering to higher resolutions, as complex or faraway objects may contain polygons whose size is much lower than the binary mask resolution. Furthermore, these methods do not tackle the problem of data complexity: The number of polygons to render remains the same regardless of the distance. An occluding polygon yields a one-operation per pixel, but no operation allows to replace a set of micropolygons with a macropolygon having the same occlusion mask. Our model combines binary masks and pre-filtered geometry.

2.3 Reflectance Pre-Filtering

Some methods store directly the directional radiance leaving a surface element in a texture, and filter on a way accounting for correlation and visibility. For instance, Ma et al. [MCT*05] encode a surface's radiance map in a BTF (Bidirectional Texture Function) for several points of view. The major drawback is that BTF's angular dimension must be densely sampled to capture the reflectance function's high frequencies and avoid ghosting artifacts. Accurate representation of highly specular functions thus become out of reach with this kind of representation for a reasonable memory budgets.

Wu et al. [WDR09] represent the details at each scale using characteristic points from the finest scale weighted with a view-dependent function. The characteristic points are chosen in such a way that the difference with the initial surface's reflectance is minimized. This approach allows the reconstruction of high frequency reflectance functions and takes masking and visibility effects into account, on the condition that an important number of view points are sampled. The underlying data structure is tedious to construct and manipulate, and is not well-suited for real-time rendering. Many characteristic points are needed to be able to reconstruct complex and high frequency effects, and may differ neighboring zones. This makes interpolation difficult or, worse, produces a discontinuous function. Furthermore, at the lowest resolution, the number of characteristic points is such that the rendering cost increases while the resulting colors and BRDFs are potentially uniform. This contradicts the expected gain of pre-filtering, for which a simple appearance should imply a limited memory cost, and expected property of multi-scale approaches, for which the rendering time per pixel should have a nearly constant complexity. To avoid these problems,

our model is based on the pre-filtering of the parameters used to analytically compute the view-dependent radiance leaving the surface.

2.4 Pre-Filtering of Surface Attributes

The local illumination equation expresses the light intensity reflected by a surface towards the observer as a function of the surface attributes.

$$I = \frac{\int_A E_{\mathbf{l}}(\mathbf{x})\ \rho(\mathbf{v},\mathbf{l},\mathbf{x},\mathbf{n}(\mathbf{x}))\ V_{\mathbf{v}}(\mathbf{x}) V_{\mathbf{l}}(\mathbf{x})\ (\mathbf{v}\cdot\mathbf{n}(\mathbf{x}))^+\ \mathrm{d}\,\mathbf{x}}{\int_A V_{\mathbf{v}}(\mathbf{x})\ (\mathbf{v}\cdot\mathbf{n}(\mathbf{x}))^+\ \mathrm{d}\,\mathbf{x}} \tag{1}$$

gives the light intensity I perceived in the direction $\mathbf{v}$ from the surface A which projects on a given pixel's footprint. At point $\mathbf{x}$ of A, $(\mathbf{v}\cdot\mathbf{n}(\mathbf{x}))^+$ is the clamped scalar product between the surface normal and the view-direction, $V_{\mathbf{v}}$ and $V_{\mathbf{l}}$ designates the visibility values along the eye and the light source and $E_{\mathbf{l}}$ the entering radiance emitted by the environment from the direction $\mathbf{l}$. We suppose that the BRDF ρ can be expressed as a sum of elementary BRDFs[3] [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).

$$\rho(\mathbf{x},\mathbf{n}(\mathbf{x})) = \sum_i a_i(\mathbf{x})\ \rho_i(\mathbf{x},\mathbf{n}(\mathbf{x})) \tag{2}$$

For simplicity, in the following we consider Eqn (1) only for the term i of Eqn (2) decomposition. These surface attributes a_i are usually stored in textures. Usual filtering techniques consider the contents of these textures as mutually uncorrelated and filter them separately (see [BN11]). The uncorrelation hypothesis between the attributes, the normals, the visibility, and the lighting allows to approximate

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

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 value of the incoming radiance, the surface attributes, the normals, and the visibility from the light source.

This is typically used for MIPmapping a color texture. This approximation was valid at the time when the resolution of the geometric details was considerably lower than the resolution of the textures. The surfaces getting more and more complex in games and CG movies, this hypothesis is no longer valid. For instance, normals produce non-linear effects, particularly for specular BRDF. Simply applying the BRDF equation to the mean normal – $\rho(\mathbf{v},\mathbf{l},\bar{\mathbf{n}})$ – does not produce the right result, and it is necessary to compute the mean BRDF $\bar{\rho}(\mathbf{v},\mathbf{l}) = \frac{\int_A \rho(\mathbf{v},\mathbf{l},\mathbf{x},\mathbf{n}(\mathbf{x}))\ \mathrm{d}\,\mathbf{x}}{\int_A \mathrm{d}\,\mathbf{x}}$ instead. Tokswig [Tok05] shows how to use the norm of MIPmapped normals to compute a correct specular exponent in the Phong model. Fournier [Fou92] introduces NDFs (Normal Distribution Functions), and describes their relationship with BRDFs and the way to filter them. Han et al. [HSRG07] use spherical harmonics to represent NDFs (which fit in Eqn (2) as a sum in the spectral domain, a_i being the coefficients associated to the harmonics).

[3] Note that in Eqn (2) $\mathbf{n}$ parameter is meant for local normals, but more generally, it can represent a large vector of any non-linear scalar or vector shader parameters attached to the surface – *e.g.,* Phong's roughness exponent.

Symbol	description
E_l	incoming radiance
ρ	BRDF
$V_{\mathbf{v}}$	visibility from eye
$V_{\mathbf{l}}$	visibility from light source
$\mathbf{n}$	local normal
a	local attribute
$\mathbf{x}$	local position
$\bar{q}$	effective average of q over the pixel
$\bar{q}(v)$	view-dependant average
$q_\bullet, \sigma_q$	either Gaussian parameters (for $q = h$ or $\mathbf{n}$) or pseudo-affine encoding (for a) of q distribution

Note that all the models above solve the BRDF filtering (*i.e.,* convolution) over the surface NDF by unifying the BRDF and NDF representations: Usual BRDF can be seen as a NDF convolved with a canonical BRDF. Filtering NDFs and filtering a BRDF over an NDF are thus a same operation.

Concerning spherical harmonics, a large number of coefficients is necessary to represent the NDF, which makes such a model extremely costly in terms of storage and computations, even for middle frequency NDFs. Furthermore, this representation does not account for masking and shadowing effects. These effects are taken into account by Tan et al. [TLQ*08] but are computed as an attenuation factor dependent on the global visible proportion of microfacets. Moreover, they are supposed separable and uncorrelated to the various components of the BRDF. Still, for real-world materials, the surface attributes which contribute to the computation of the radiance can be highly correlated to their visibility. Wu et al. [WDR11] propose a way to compute Eqn (1) by introducing a BVNDF (Bidirectional Visible Normal Distribution Function). Their algorithm discretizes this function over the hemisphere, making precomputation and store complex. In the logic of saving memory and computational time – which is essential for real-time – we instead favor a customizable analytical model to represent microgeometry rather than pre-computing and storing it in a "brute force" way. Our results may be less exact in complex cases, but our real-time model produces similar view-dependent effects in most cases with a much easier and lighter implementation.

2.5 Microscopic Surface Models

Analytical BRDF models usually reproduce the reflectance of surfaces of known statistical properties. Microfacets-based BRDFs models [CT81, ON94] match surfaces with Gaussian microgeometry. Masking $V_{\mathbf{v}}$ and shadowing $V_{\mathbf{l}}$ are represented through the multiplication of the occlusion-free BRDF ρ_0 with a geometrical attenuation factor G representing each microfacet's visible and lit proportion. This model has the inconvenient of computing only the self-shadowing of a facet. In addition, on real-world surfaces (such as terrain, cloth material, crackled rusty or embossed material, etc.) the correlation of an attribute with its visibility often derives from its correlation with its depth h within the surface. This cannot be modeled with the visibility along a microfacet. Thus, Smith's model [Smi67] which gives an analytical formula for computing the probability

of shadowing of a point on a Gaussian surface as a function of h is more suited to take these correlations into account. For a surface where the depth $h(\mathbf{x})$ and the normal $\mathbf{n}(\mathbf{x}) = (n_x, n_y)$ are two Gaussian random processes $\mathcal{N}(0, \sigma_h)$ and $\mathcal{N}(0, (\sigma_{n_x}, \sigma_{n_y}))$, then the probability of visibility $V(\mathbf{v}, h, n_x, n_y)$ is given by [Smi67]:

$$V(\mathbf{v}, h, n_x, n_y) = V(\mathbf{v}, h)V(\mathbf{v}, n_x, n_y) \tag{4}$$

where

$$V(\mathbf{v}, n) = \text{heaviside}(\mu(\mathbf{v}) - n) \tag{5}$$

$$V(\mathbf{v}, h) = \left[1 - \frac{1}{2}\,\text{erfc}(\frac{h}{\sqrt{2}\sigma_h})\right]^{\Lambda(\mathbf{v})} \tag{6}$$

with

$$\Lambda(\mathbf{v}) = \frac{1}{2}\left(\sqrt{\frac{2}{\pi}}\frac{\sigma_n(\mathbf{v})}{\mu(\mathbf{v})}e^{-\frac{\mu(\mathbf{v})^2}{2\sigma_n(\mathbf{v})^2}} - \text{erfc}\left(\frac{\mu(\mathbf{v})}{\sqrt{2}\sigma_n(\mathbf{v})}\right)\right) \tag{7}$$

$\mu(\mathbf{v}) = \cot(\theta)$ 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 $\mathbf{v}$. This modelization enables the expression of the visibility as a function of the depth h independently of the normal. In our model, we use this formulation to compute the visibility of an attribute correlated to its depth in the surface.

Note that the representation of BRDF trough slope statistics provides another way to unify the NDF and the BRDF in order to facilitate their convolution (especially with Gaussian statistics).

3 Our General Rendering Framework

Our rendering model integrates the radiance reaching the pixel by using a volumetric cone tracing through a hierarchical representation of the view-dependent pre-integrated geometry. A cone is defined as a set of successive cone elements (Figure 3 (a)) 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-rays cones are treated similarly, launched from contributing cone elements, and sized so as to fit to cone element and to light source diameters. By pre-computing a hierarchy of interpolable neighborhoods, we use the local cone diameter to access the right MIPmap level. Thereby, our rendering scheme is similar to volume rendering with differential cones [CNLE09], but our storage and shading of voxels accounts 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.

This section begins with a formal introduction to our cone tracing (3.1), our differential cone tracing model (3.2) and its components : the models of local illumination (3.3) and visibility (3.4). In section 4, we propose an effective computation model based on a representation of microgeometry and surface attributes distributions.

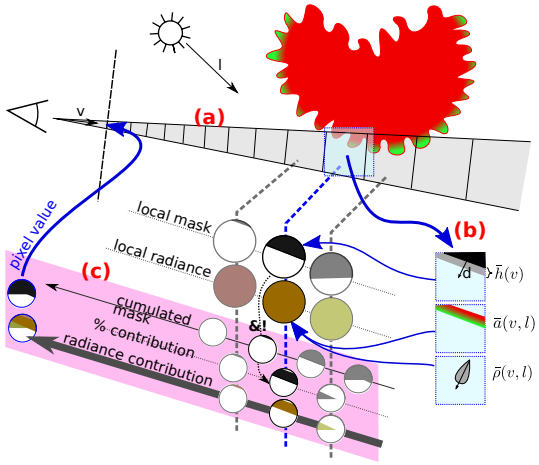


Figure 3: *Algorithm and representation overview. (a):* We consider successive elements along the ray cone. *(b):* In each element, we sample one view-dependent pre-filtered information. It consists of the view dependent average silhouette plane $d + \bar{h}(\mathbf{v})$, the light-view dependent average attributes $\bar{a}(\mathbf{v}, \mathbf{l})$, and the light-view dependent average BRDF $\bar{\rho}(\mathbf{v}, \mathbf{l})$. *(c):* To get the pixel value, the fragments' contributions to each cone element must be recomposed accounting for occlusion correlation. In our computational model, we rely on Carpenter subpixel masks [Car84].

3.1 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 visibility of the geometry A for the ray ω is a binary occlusion value $V(\omega) \in \{0, 1\}$. The light intensity I perceived on the pixel, and reflected by the geometry of the scene, is the sum of the outgoing radiances L towards the viewer's direction

$$I = \int_{\Omega} V(\omega) L(\omega) \ \mathrm{d}\,\omega \tag{8}$$

This equation describes I as the result of a perfect sampling over the pixel footprint *i.e.,* the mean radiance over an infinity of directions ω with infinitely small solid angles $\mathrm{d}\,\omega$. This sampling of the scene can be highly non-local (*e.g.,* if non-connected fragments of the geometry are visible through the same pixel). A more localized description of this integral expresses it as the sum of the accumulated radiances through the space covered by the cone

$$I = \int_{0}^{\infty} \int_{S(z)} V(\mathbf{x}_{\omega,z}) L(\mathbf{x}_{\omega,z}) \ \mathrm{d}\,\mathbf{x}_{\omega,z} \ \mathrm{d}\,z \tag{9}$$

where, for each cone section $S(z)$ at a distance z, the point $\mathbf{x}_{\omega,z}$ is the intersection of $S(z)$ with the ray associated to the direction ω. $L(\mathbf{x}_{\omega,z})$ is the outgoing radiance of the coincident surface $\{ \mathbf{x}_{\omega,z} \mid V(\mathbf{x}_{\omega,z}) = 1\}$ at the intersection of

the visible geometry and the cone section. We write the occlusion produced by the geometry with the indicator functions

$$V(\mathbf{x}_{\omega,z}) = \mathbb{1}_A(\mathbf{x}_{\omega,z})\ (1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z[})) \tag{10}$$

where $\mathbb{1}_A(\mathbf{x}_{\omega,z})$ is the binary occlusion value indicating the intersection of the ray and the geometry A at the point $\mathbf{x}_{\omega,z}$. The multiplication with the indicator function $1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z[})$ expresses the fact that only visible points on the section $S(z)$ contributes to the mean radiance at that distance.

3.2 Differential Cone Tracing

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

$$I = \sum_{i=0}^{\infty} \int_{z_i}^{z_{i+1}} \int_{S(z)} V(\mathbf{x}_{\omega,z}) L(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z \tag{11}$$

To permit pre-filtering, our objective is to find a way to represent the pre-integrated local visibility $V_i = \int_{z_i}^{z_{i+1}} \int_{S(z)} V(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z$ and the mean local visible outgoing radiance $L_i = \frac{\int_{z_i}^{z_{i+1}} \int_{S(z)} V(\mathbf{x}_{\omega,z}) L(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z}{\int_{z_i}^{z_{i+1}} \int_{S(z)} V(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z}$ in a cone element. Then, at runtime we only need to compute $I = \sum_{i=0}^{\infty} V_i L_i$.

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

3.3 Outgoing Radiance L_i in a Cone Element

We make the hypothesis **$(\mathbf{H_1})$** that correlation between radiance and visibility only exists at the neighborhood's scale, and that there is no correlation between long-distance occlusion and local radiance. This allows us to consider local L_i independently. We have $L_i \approx \frac{\int_{z_i}^{z_{i+1}} \int_{S(z)} \mathbb{1}_A(\mathbf{x}_{\omega,z})\ L(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z}{\int_{z_i}^{z_{i+1}} \int_{S(z)} \mathbb{1}_A(\mathbf{x}_{\omega,z})\ \mathrm{d}\,\mathbf{x}_{\omega,z}\ \mathrm{d}\,z}$ by canceling the term $1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z[})$ which gets out of the integral thanks to uncorrelation hypothesis.

To compute L_i,we need a model which describes the geometry inside the i^{th} cone element, a model for the distribution of the surface attributes (we propose one in section 4), and the analytical integration of the masking and shadowing effects on theseattributes over a complex surface (Figure 3 (b)). By considering the correlation between the surface attributes and their visibility, we get a cousin form of Eqn (3)

$$L_i \approx \bar{E}_{\mathbf{l}}\ \bar{a}_i(\mathbf{v},\mathbf{l})\ \bar{\rho}_i(\mathbf{v},\mathbf{l})\ \bar{V}_{\mathbf{l}} \tag{12}$$

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

$$\bar{a}_i(\mathbf{v},\mathbf{l}) = \frac{\int_A a_i(\mathbf{x})\ V_{\mathbf{v}}(\mathbf{x}) V_{\mathbf{l}}(\mathbf{x})\ (\mathbf{v}\cdot\mathbf{n}(\mathbf{x}))^+\ \mathrm{d}\,\mathbf{x}}{\int_A V_{\mathbf{v}}(\mathbf{x}) V_{\mathbf{l}}(\mathbf{x})\ (\mathbf{v}\cdot\mathbf{n}(\mathbf{x}))^+\ \mathrm{d}\,\mathbf{x}} \tag{13}$$

From Eqn (3), we only keep the earlier hypothesis of long distance uncorrelation between radiance and occlusion (which allows to take $E_{\mathbf{l}}$ out of the integral (1)) – this is already in **$(\mathbf{H_1})$** – , and the hypothesis of uncorrelation between $a_i(\mathbf{x})$ and $\rho_i(\mathbf{x},\mathbf{n}(\mathbf{x}))$ – let's denote it **$(\mathbf{H_2})$**.

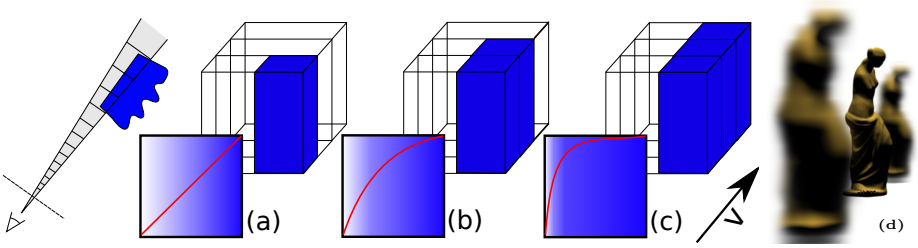


Figure 4: *(a):* A pixel is only half covered by opaque geometry, thus the fragment has an opacity $\alpha = 0.5$. *(b,c):* Successively accumulating opacity by naive blending progressively saturates the final result, while it should stick to 0.5. This tends to thicken silhouettes of antialiased or unfocused shapes *(d). I.e.,* the α-blending model is wrong when fragments (or successive cone elements) are highly correlated, which is the case at silhouettes.

3.4 Visibility V_i in a Cone Element

The visibility of the i^{th} cone element

$$V_i = \int_{z_i}^{z_{i+1}} \int_{S(z)} \mathbb{1}_A(\mathbf{x}_{\omega,z}) \, (1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z[})) \, \mathrm{d}\,\mathbf{x}_{\omega,z} \, \mathrm{d}\,z \tag{14}$$

can be rewritten as

$$V_i = \int_{\Omega} \mathbb{1}_A(\mathbf{x}_{\omega,[z_i,z_{i+1}]}) \, (1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z_i]})) \, \mathrm{d}\,\omega \tag{15}$$

where $\mathbb{1}_A(\mathbf{x}_{\omega,[z_i,z_{i+1}]})$ is the indicator function of the intersection of the ray going through $\mathbf{x}$ and the surface A in the cone element $[z_i, z_{i+1}]$. 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

$$V_i = \int_{\Omega} \mathbb{1}_A(\mathbf{x}_{\omega,[z_i,z_{i+1}]}) \, \mathrm{d}\,\omega \int_{\Omega} (1 - \mathbb{1}_A(\mathbf{x}_{\omega,[0,z_i]})) \, \mathrm{d}\,\omega \tag{16}$$

which corresponds to the blending model $\alpha_i = \alpha_{[z_i,z_{i+1}]}(1 - \alpha_{i-1})$ 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 (cf Figure 4). A good rendering model should thus take the correlation between the terms in the integral (15) into account. Evaluating V_i requires to represent and to manipulate the functions $\mathbb{1}_A(\mathbf{x}_{\omega,[z_i,z_{i+1}]})$.

4 Our Computation Model

In this section, we propose a way to represent the microgeometry and the attributes distributions in order to calculate Eqn (13) and (15).

4.1 Hypothesis

We base our approach on five additional hypotheses :
($\mathbf{H_3}$) The microgeometry is assimilated to a Gaussian surface, 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.
($\mathbf{H_4}$) The BRDF $\rho_i(\mathbf{x}, \mathbf{n}(\mathbf{x}))$ and the depth h of the surface are uncorrelated, and in particular normals in respect to depth (for applyability of [Smi67]. But it is already a consequence of **($\mathbf{H_3}$)**).
($\mathbf{H_5}$) We assume the surface attributes a_i correlated only to their depths h within the surface (which is the case for numerous real surfaces). This allows to separate $\int a(\mathbf{x})$ and $\int \mathbf{n}(\mathbf{x})$ in Eqn (13).
($\mathbf{H_{5bis}}$) We assume that the distributions of the average attributes values with depth can be represented as a gradient between the depths and heights of the surface details: $a(h) = \bar{a}_\bullet + \sigma_a g(h)$, where $\bar{a}_\bullet$ is the mean value of the attribute and $g(h)$ a normalized increasing function. It is interesting to take a sigmoid function as g to avoid spoiling the dynamics of a in loosely representative extrema. We choose $g(h) = 1 - \frac{1}{2}\operatorname{erfc}\left(\frac{h}{\sqrt{2}\sigma_h}\right)$, which enables the analytical integration of Eqn (13). The parameter σ_a represents the correlation between h and a_i. This computation is detailed in section 4.3.
($\mathbf{H_6}$) 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 the visibility V is also based on that approximation. This hypothesis fixes the validity domain of our model: We do not handle the "ultimate filtering" case which occurs when a large part of the object projects into one single pixel.

4.2 Overview

We now explain how we use these hypotheses to achieve the practical precomputation of a multi-scale surface representation, and the practical calculation of illumination and visibility at runtime.

Computation of Local Visibility. We represent the functions V_i of Eqn (15) with binary masks [Car84] computed for each cone element (Figure 3 (c)). It enables us to compute the contribution of the local geometry to the pixel, while taking into account the correlations with occlusions upstream along the cone's axis. To compute this mask, the geometry is locally approximated by a fixed plane $\bar{h}_\bullet$ modulated with an offset computed from the view-dependent microscopic occlusion effects (see section 4.4). This plane defines a half-space whose 3D intersection with the cone element gives a 2D occupancy distribution over the pixel. We associate a tabulated mask to that distribution.

BRDF Representation. To ease the convolution of NDFs with BRDFs, we assume as in previous work that both can be represented the same way. We rely on their Gaussian slope statistics representation [CT81, ON94]: the initial microfacet statistics is progressively enriched by the filtering of meso-surface normals. Still, our scheme is compatible with lobe-based representations [Fou92, Tok05, HSRG07] as well.

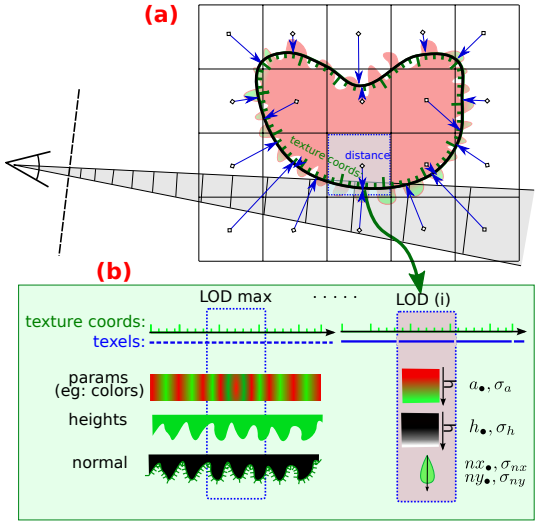


Figure 5: The low-frequency "blurry geometry" is proxied via a coarse volumetric grid *(a)* providing surface location through a distance field and index to all details (stored in MIPmapped textures *(b)*) through texture coordinates. Data for a cone element are obtained interpolating the grid data then MIPmapping in the texture.

Voxel-based Data Structure Representation. We consider two alternate data structure implementations: one for volume-based data, and one for surface-based data. Calculations are identical and performances quite similar.

We consider volumetric objects as signed distance fields $\mathbf{d}$ stored in octree (value and gradient). Details relative to the coarse surface are assumed to respect a Gaussian statistics. The coarse surface is assumed to be not too curved within a single cone element, since our integrations do not account for curvature. Our filtering is inaccurate at distance passed the limit above (but still better that any non-oversampling method). In addition to $\mathbf{d}$, each voxel stores filtered geometric information as $\bar{h}_\bullet$ and σ_h; normal filtered information as $\bar{\mathbf{n}}_\bullet$ and $\sigma_{n_x}, \sigma_{n_y}$; and each attribute a_i as $\bar{a_i}_\bullet$ and σ_{a_i}. Each parameter is initialized from the corresponding input data v as $\bar{v}_\bullet = v$ and $\sigma_v = 0$. Using RGB color attribute as a, this makes 15 values per voxel in total (possibly only 9 at the deepest octree level). Our rendering scheme is thus an extension of [CNLE09] with the addition of sub-voxel effects.

Since full volume representations are not reasonable memory-wise for many CG applications, we also propose an alternate surface-based data structure implementation which is way more memory-efficient.

Surface-based Data Structure Representation. (see Figure 5). Here, the input data assumes the form of a coarse mesh, "pseudo-volumetric" details, texture maps for attributes a_i (such as colors), and the corresponding mapping $\mathbf{t} = (t_x, t_y)$. Pseudo-volumetric details can be an height map, a parallax map, a volumetric texture or shell map, or an hyper-texture shader, as long as the Gaussian surface statistics is reasonably obeyed. If only a very detailed mesh is

available, it can be converted into coarse mesh along with a height field using [COM98].

Our cone-tracing requires a volumetric representation to be traversed in the surface vicinity in the spirit of [TNF89] blurry layer. While some of the aforementioned pseudo-volumetric structures (*e.g.,* shellmap) could be used directly, our implementation relies on a simple coarse 3D-grid voxelization around the mesh. Our voxels contain the distance field $\mathbf{d}$ to the coarse mesh sampled at the voxel centers (signed value and gradient), and a copy of the texture coordinates at the pointed surface location. This way, we can emulate a detailed content in each coarse voxel by modulating the distance field with the stored detailed $h(\mathbf{t}(\mathbf{x}))$. Differently to our volume-based implementation above, here we thus do not access an explicit octree at the LOD controlled by the cone width. Instead, the content of the cone element is obtained by accessing 2D maps with the corresponding LOD. The coarse surface is assumed to be not too curved within the voxel span, so that the trilinear interpolation of the distance field correctly represents the surface. The voxel layer around the surface must be thick enough to account for large cones (for distant or out of focus views). Since we do not handle the ultimate filtering case, where cone elements engulf large object parts, such cases can be treated approximately, either by limiting the maximum LOD level or by using an octree hierarchy with the crude quadrilinear filtering of $\mathbf{d}$ (which is still better than what any non-oversampling method can do).

We pre-compute a hierarchical representation of filtered attributes and geometric details. We rely on the same mapping indexation as before to store all our data in MIPmapped 2D maps: this emulates a 3D field without the storage cost of volume data. Unlike ordinary MIPmap, we compute each level with proper filtering algorithms (see sections 4.3 and 4.4). We store the geometry filtered information as $\bar{h}_\bullet$ and σ_h; the normal filtered information as $\bar{\mathbf{n}}_\bullet$ and $\sigma_{n_x}, \sigma_{n_y}$; and each attribute a_i as $\bar{a_i}_\bullet$ and σ_{a_i}. Each parameter is initialized from the corresponding input data v as $\bar{v}_\bullet = v$ and $\sigma_v = 0$. The norms of the space-to-texture Jacobians must also be stored to determine the proper LOD to use. If a represents a color, this makes a total of 13 texture channels.

4.3 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 view-dependent mean surface attributes $\bar{a}(\mathbf{v}, \mathbf{l})$ involved in the computation of the outgoing radiance (Eqn (12)).

According to **$(\mathbf{H_5})$**, the attribute $a(h)$ and the visibility values $V(\mathbf{v}, h)$ (from the eyes) and $V(\mathbf{l}, h)$ (from light source) are expressed as functions of h. We reformulate Eqn (13) by integrating over h:

$$\bar{a}_\bullet(\mathbf{v}, \mathbf{l}) = \frac{\int_{-\infty}^{\infty} a(h)\ V(\mathbf{v}, h) V(\mathbf{l}, h)\ P(h)\, \mathrm{d}\, h}{\int_{-\infty}^{\infty} V(\mathbf{v}, h) V(\mathbf{l}, h)\ P(h)\, \mathrm{d}\, h} \tag{17}$$

where a surface element has a probability $P(h, \vec{bn}) \sim \mathcal{N}(0, \sigma_h)$ to be at depth h, has an attribute $a(h)$ (average of $a(p)$ over points p of depth h), and has probability of visibility $V(\mathbf{d}, h)$ given by Smith's model (Eqn (6)) for a direction $\mathbf{d}$.

In Eqn (17), we can expand $a(h)$ out of the integral. According to Smith's

model, we have $P(h) = g'(h)$ and $V(\mathbf{v}, h) = g(h)^{\Lambda(\mathbf{v})}$. Eqn (17) hence becomes

$$\bar{a}(\mathbf{v}, \mathbf{l}) = \bar{a}_\bullet + \sigma_a \frac{\displaystyle\int_{-\infty}^{\infty} g'(h)\ g(h)^{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})+1}\ \mathrm{d}\,h}{\displaystyle\int_{-\infty}^{\infty} g'(h)\ g(h)^{\Lambda(\mathbf{v})+\Lambda(\mathbf{l})}\ \mathrm{d}\,h} \tag{18}$$

which has the following analytical solution

$$\bar{a}(\mathbf{v}, \mathbf{l}) = \bar{a}_\bullet + \sigma_a \frac{\Lambda(\mathbf{v}) + \Lambda(\mathbf{l}) + 1}{\Lambda(\mathbf{v}) + \Lambda(\mathbf{l}) + 2} \tag{19}$$

4.4 Computation of the Local Visibility

This section proposes a representation and an algorithm to compute $\mathbb{1}_A(\mathbf{x}_{\omega,[z_i,z_{i+1}]})$ necessary for the evaluation of Eqn (15). According to hypothesis **($\mathbf{H_{5bis}}$)**, the mean geometry can be locally represented by the plane specified by the signed distance d enhanced with a local mean variation $\bar{h}_\bullet$, and the normal $\frac{\mathbf{d}}{|\mathbf{d}|} = (\delta_x, \delta_y, \delta_z)$. The intersection of this plane and a cone element is computed analytically. We represent the functions $\mathbb{1}_A$ as binary masks distributed over the pixel footprint. We can thus compute and combine them efficiently as in [Car84].

Computation of the Mean View-Dependent Visible Plane. The local geometry fluctuates around the fixed plane $\bar{h}_\bullet$ and the offset between this plane and the silhouette is view-dependent (see Figure 6). To compute an accurate cone element occlusion mask, we first need to adjust the mean plane accordingly. To evaluate the correct view-dependent silhouette $d(\mathbf{v})$, we have to compute the mean visible depth $\bar{h}(\mathbf{v})$ relative to the coarse surface: $d(\mathbf{v}) = d + \bar{h}(\mathbf{v})$. $d(\mathbf{v})$ is given by an equation analog to the one used for computing $\bar{a}(\mathbf{v}, \mathbf{l})$:

$$\bar{h}(\mathbf{v}) = \bar{h}_\bullet + \frac{\int_{-\infty}^{\infty} h\ V(\mathbf{v}, h)\ P(h)\,\mathrm{d}\,h}{\int_{-\infty}^{\infty} V(\mathbf{v}, h)\ P(h)\,\mathrm{d}\,h} \tag{20}$$

This integral has no analytical solution. However, it is linearly dependent of σ_h and of a function of $\Lambda(\mathbf{v})$. We use the approximation $\bar{h}(\mathbf{v}) = \bar{h}_\bullet + 0.39\sigma_h \log(1 + 4.75\Lambda(\mathbf{v}))$ which ensures a maximum error of 14% over $\bar{h}(\mathbf{v})$.

Computation of the Binary Masks. We use binary masks, which are pre-computed and stored in a bidimensional table as in [Car84]. To index the masks, Carpenter uses the residuals of the rasterization of the 2D polygons edges by Bresenham's algorithm taken at the bottom and top of the pixels sides. Here, we start directly from the location of the 3D plane $d(\mathbf{v})$ within the cone element. The function $\mathbb{1}_A(p_{\omega(x,y),[z_i,z_{i+1}]})$ locally only depends on two parameters θ and v (see Figure 7). The appendix B explains how to compute these parameters from $\mathbf{d}(\mathbf{v})$. The binary masks are pre-computed and stored in a bidimensional table indexed by θ and v at the runtime. The binary operations $\cup$, $\cap$ and bitcount [Car84] allow the evaluation and update of V_i in Eqn (15) for each iteration i during the cone tracing in an efficient and simple way.

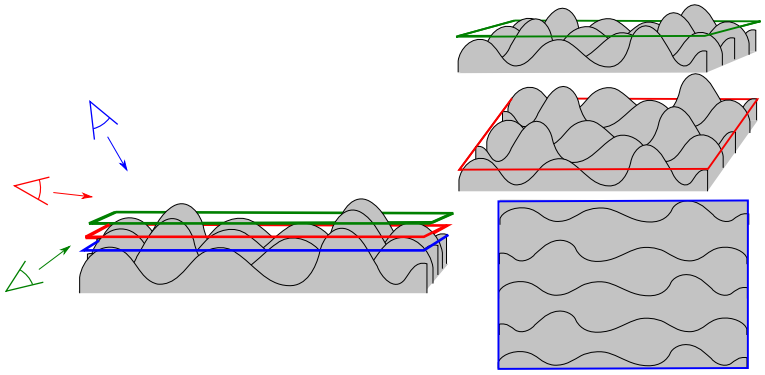


Figure 6: The mean visible depth – which settles the average masking plane in a cone element – is view-dependent.

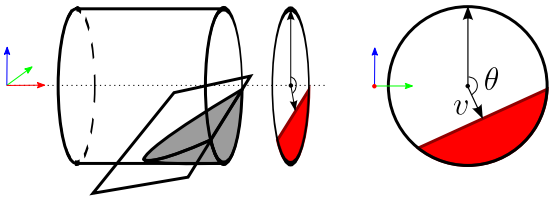


Figure 7: The mask represents the projection of the 3D intersection of the geometry and the cone element.

5 Implementation and Results

We implemented our algorithm on a dual core Intel with an nVidia GTX560 graphics card. In all our examples, the volumetric texture containing the distance field is 64^3 and requires 4MB storage (without any optimization such as sparse representation). The details are stored in 1024×1024 MIPmapped 2D textures with 10 to 13 channels, which form a total of about 8MB. To allow really deep zooms, we further enhance these details with 3 to 8 octaves of 3D Perlin noise: The close views of Figure (1b) and (1c) have virtually a resolution of 8192^3. Our filterable BRDF relies on microBRDF slope statistics [ON94, CT81] progressively enriched by the noise NDF then the details NDF. The pre-computation time is 2 secs. We use masks with 128 Poisson-distributed samples, so that atomic mask operations are done using four 32bits integers. Our precalculed mask table is 256×256.

All 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). When zooming, the cost per covered pixel is nearly constant around 0.1-0.3 μs/pixel. The cost mainly depends on the silhouettes: views with no silhouettes are the fastest, views with large grazing areas are the costliest since several cone elements per ray are computed.

View Fig 9.n	far view .1	mid view .2	close view .3	closer (silh) .4	closer (no silh)	$view_1$ DoF($\varnothing$10) .1	$view_1$ shadow .1
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

Our algorithm is able to correctly reproduce subpixel color effects due to correlated visibility from the eyes and from the light source, that are comparable to the ground truth, contrary to the naive method processing separate filtering of maps (see Figure 8). In particular, we ensure seamless zooms with close to no color shift (see Figure 10) and correct transformation of meso-granularity to BRDF roughness (see Figure 9, while keeping good real-time performances. See also the companion video.

Our model deal with anisotropy: on Figure 1(d), the color in the left area shifts from green (top-left) to red when the light is tilted horizontally (top-right) and to dark-green when it is tilted vertically (bottom-left). Our model can also be used directly as a material editor without the burden of managing explicit details (see Figure 11). In such case the shader is concise and easy to insert in an existing rendering pipeline (see Appendix A).

Also, 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 4(d) for cone rendering on volumes). Indeed, our scheme works exactly the same for depth of field, yielding even better performances: As for [CNLE09], our cone-tracing scheme is faster for soft shadows and depth of field (see Figure 11) than for sharp shadows and focussed images, as the former relies on coarser LOD.

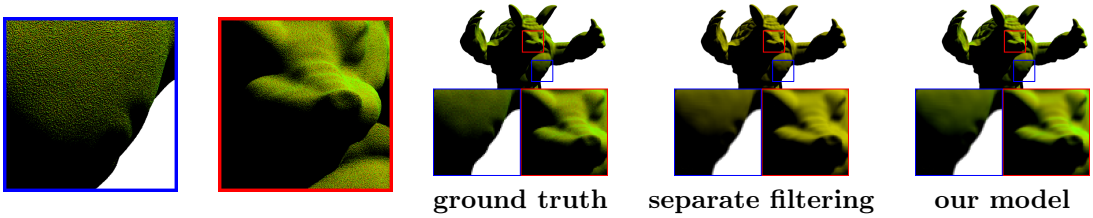


Figure 8: *Comparisons of lightview-dependent color effects (right).* Grazing light or view directions cancels the contribution of colors correlated to deep locations (here, the red) as seen in the 2 regions of interest *(left)*: Average color shifts from yellow to green. Naive separate filtering of colormap gives uniform yellow, while our model reproduces the ground truth.

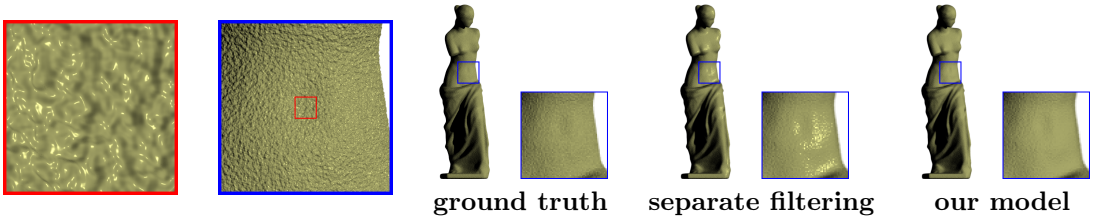


Figure 9: *Comparisons of emboss-to-shading filtering (right).* A bumpy specular area (see regions of interest on *left*) appears diffuse at distance: with a correct filtering, details go from geometry to BRDF. Naive separate filtering of normals applied to the base BRDF gives the wrong shading, while our model reproduces the ground truth.

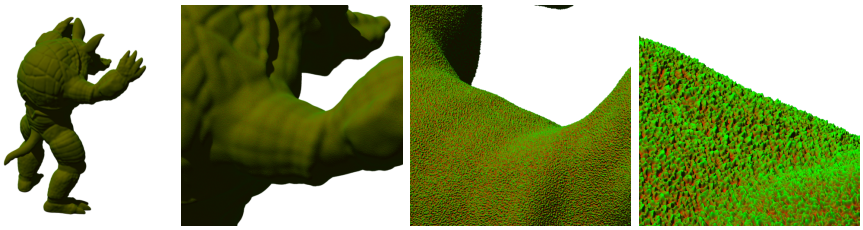


Figure 10: Our model shows seamless transitions at zooming (see also the video).

Figure 11: *Left:* Our method enables real-time anti-aliasing, depth of field, and soft shadows, without oversampling. *Right:* Our representation can be used directly as a material editor. The user provides a depth-wise color gradient and a slope statistics σ_n, possibly anisotropic ($\sigma_{n_x}, \sigma_{n_y}$ along the two texture mapping axis).

6 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 zooms, and maintains coherency through scales while achieving real-time results. We produce antialiased constant-cost notably accurate effects in real-time, making the management of very detailed objects scalable without compromising quality or performances. We have shown that our model can also be used as a simple view-dependent material editor easily integrable in an existing pipeline.

Our contributions are two folds: a theoretical framework, and a computational model with stronger practical hypothesis. It could be possible to extend or revisit several of our hypotheses, such the modelization of depth-dependent attributes, and [TLQ*08]-like min-max combinations of $\Lambda(\mathbf{v})$ and $\Lambda(\mathbf{l})$ could be introduced in Eqn (19) to simulate hotspot effects. Also, our "proof of concept" implementation could be done in very different ways: as mentioned before, one could rely on existing pseudo-volumetric structures such as shellmaps, skinning-like representations being especially interesting in the scope on animation. Moreover, a texture-mapping based bulletproof implementation should handle important details such as discontinuity of texture coordinates more carefully. Another approximation is that we assumed that the large numbers law was always valid for our statistical representations, which is not the case during transitions (as shown in the slight color shift in the zoom): In such situations, limited supersampling of the cone element should be done as long as the lowest wavelength of its content is close to its characteristic size.

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 non-valid or restrictive hypothesis of common pre-filtering schemes as possible. Still, more complex configurations exist, and deep-filtering remains a Grail – starting with accounting for the curvature of coarse surface. 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 distance (grass, wire mesh, foliage, semi-transparent material, etc.). Our volume implementation assumed opaque objects with defined coarse surface. Adapting it to the filtering of view-dependent effects in semi-transparent volumes would be another interesting but challenging future work.

References

[Ama84] Amanatides J.: Ray tracing with cones. In *Proceedings of SIGGRAPH '84* (1984), pp. 129–135.

[AW85] Abram G., Westover L.: Efficient alias-free rendering using bit-masks and look-up tables. In *Proceedings of SIGGRAPH '85* (1985), pp. 53–59.

[BM93] Becker B. G., Max N. L.: Smooth transitions between bump rendering algorithms. In *Proceedings of SIGGRAPH '93* (1993), pp. 183–190.

[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).

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

[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).

[COM98] Cohen J., Olano M., Manocha D.: Appearance-preserving simplification. In *Proceedings of SIGGRAPH '98* (1998), pp. 115–122.

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

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

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

[HH84] Heckbert P. S., Hanrahan P.: Beam tracing polygonal objects. In *Proceedings of SIGGRAPH '84* (1984), pp. 119–127.

[Hop97] Hoppe H.: View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH '97* (1997), pp. 189–198.

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

[Kaj85] Kajiya J. T.: Anisotropic reflection models. In *Proceedings of SIGGRAPH '85* (1985), pp. 15–21.

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

[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.

[MCT*05] Ma W.-C., Chao S.-H., Tseng Y.-T., Chuang Y.-Y., Chang C.-F., Chen B.-Y., Ouhyoung M.: Level-of-detail representation of bidirectional texture functions for real-time rendering. In *Proceedings of I3D '05* (2005), pp. 187–194.

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

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

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

[Sch91] Schilling A.: A new simple and efficient antialiasing with subpixel masks. In *Proceedings of SIGGRAPH '91* (1991), pp. 133–141.

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

[TLQ*08] Tan P., Lin S., Quan L., Guo B., Shum H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics 14* (March 2008), 412–425.

[TNF89] Thomas D., Netravali A. N., Fox D.: Anti-aliased ray tracing with covers. *Computer Graphics Forum 8*, 4 (1989), 325–336.

[Tok05] Toksvig M.: Mipmapping normal maps. *JGT: journal of graphics, GPU, and game tools 10*, 3 (2005), 65–71.

[WDR09] Wu H., Dorsey J., Rushmeier H.: Characteristic point maps. *Computer Graphics Forum 28*, 4 (2009), 1227–1236.

[WDR11] Wu H., Dorsey J., Rushmeier H.: Physically-based interactive bi-scale material design. *ACM Trans. Graphic. 30*, 6 (2011).

[Wil83] Williams L.: Pyramidal parametrics. In *Proceedings of SIGGRAPH '83* (1983), pp. 1–11.

Appendices

A Shader Code

This simple shader can be used to compute the view-dependent color effects shown in figure 11 (right).

```
uniform float sigma_n;
uniform float a0;
uniform float sigma_a;

uniform vec3 color1;
uniform vec3 color2;

float Lambda(float theta, float sigma_n)
{
        float nu = cos(theta) / sin(theta);
        float x =
        sqrt(2.0/pi) *
        sigma_n/nu   *
        exp(-pow(nu/sigma_n, 2.0)/2.0);
        float y = erfc(nu/(sqrt(2.0)*sigma_n));
        return 0.5 * (x-y);
}

vec3 viewdepColor(float theta_V, float theta_L)
{
        float Lambda_V = Lambda(theta_V, sigma_n);
        float Lambda_L = Lambda(theta_L, sigma_n);
        float a =
        (a0 - sigma_a/2.0) +
        sigma_a            *
        (1.0+Lambda_V+Lambda_L)/
        (2.0+Lambda_V+Lambda_L);
        return a*color1 + (1.0-a)*color2;
}
```

B Computation of the mask parameters

The cone element i, locally approximated with a cylinder, is at distance z_i from the eyes, is oriented in direction $\mathbf{z}$, and has a width $c(z_i)$. To compute a bit $\omega(x, y)$ of the binary mask, we have to determine if the line starting from $(x, y, 0)$ and parallel to the $\mathbf{z}$ axis intersects the plane tangent to the geometry (see Figure 12). This plane of normal $(\delta_x, \delta_y, \delta_z)$ and distance $d(\mathbf{v})$ is described by the equation

$$x\delta_x + y\delta_y + (z - z_i)\delta_z + d(\mathbf{v}) = 0$$

The point $p(z_i + d_z) = (c(z_i)x, c(z_i)y, z_i + d_z)$ in the cone element is occluded by the geometry if it is in the half-space defined by the tangent plane

$$c(z_i)x\delta_x + c(z_i)y\delta_y + d_z\delta_z + d(\mathbf{v}) \leq 0$$

The ray passing through $p(z_i + d_z)$ intersects the geometry in this cone element if one of the farthest points $p(z_i \pm \frac{z_{i+1}-z_i}{2})$ to the center of the cone element is occluded. The intersection test for the bit of the mask associated to the point (x, y) of the pixel is then

$$x\delta_x + y\delta_y \leq \frac{\delta_z \frac{z_{i+1}-z_i}{2} - d(\mathbf{v})}{c(z_i)}$$

We rewrite the projection of the normal on the pixel footprint

$$(\delta_x, \delta_y) = r(\cos\theta, \sin\theta)$$

in polar coordinates and the final intersection test has the form presented in Figure 7 :

$$x\cos\theta + y\sin\theta \leq v$$

with $v = \frac{\delta_z \frac{z_{i+1}-z_i}{2} - d(\mathbf{v})}{c(z_i)\sqrt{\delta_x^2+\delta_y^2}}$.

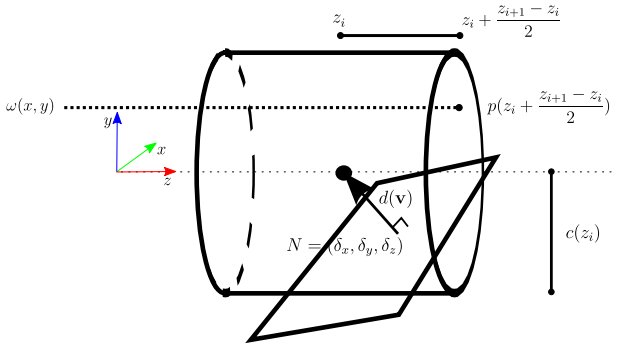
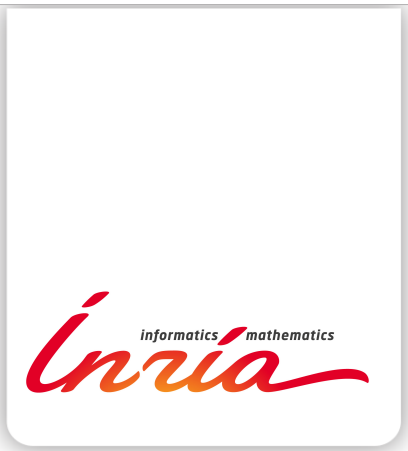


Figure 12: The bit corresponding to $\omega(x, y)$ is set to 0 if one of the points $p(z_i \pm \frac{z_{i+1}-z_i}{2})$ is below the oriented cutting plane.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399