A Survey of Non-linear Pre-filtering Methods for Efficient and Accurate Surface Shading

Eric Bruneton and Fabrice Neyret

Abstract—Rendering a complex surface accurately and without aliasing requires the evaluation of an integral for each pixel, namely a weighted average of the outgoing radiance over the pixel footprint on the surface. The outgoing radiance is itself given by a local illumination equation as a function of the incident radiance and of the surface properties. Computing all this numerically during rendering can be extremely costly. For efficiency, especially for real-time rendering, it is necessary to use precomputations. When the fine scale surface geometry, reflectance and illumination properties are specified with maps on a coarse mesh (such as color maps, normal maps, horizon maps or shadow maps), a frequently used simple idea is to *pre-filter* each map linearly and separately. The averaged outgoing radiance, *i.e.*, the average of the values given by the local illumination equation is then estimated by applying this equation to the averaged surface parameters. But this is really not accurate because this equation is non-linear, due to self-occlusions, self-shadowing, non-linear reflectance functions, etc. Some methods use more complex pre-filtering algorithms to cope with these non-linear effects. This paper is a survey of these methods. We start with a general presentation of the problem of pre-filtering complex surfaces. We then present and classify the existing methods according to the approximations they make to tackle this difficult problem. Finally, an analysis of these methods allows us to highlight some generic tools to pre-filter maps used in non-linear functions, and to identify open issues to address the general problem.

Index Terms—Rendering, anti-aliasing, pre-filtering, reflectance, surface

1 Introduction

PHOTOREALISM has always been a major goal of Computer Graphics. Local and global illumination models, corresponding respectively to complex materials and complex light transports, have been extensively studied. Today, a key problem is the management of details: highly detailed objects and very large scenes are prone to aliasing and popping, or to unreasonable rendering costs. Rendering details accurately and efficiently is thus the new challenge, not only for real-time applications, but also for CG feature films (where each frame takes hours to render).

A first step to manage these details is to represent the fine scale features of each object with some maps, defined on a coarse object mesh and used as parameters in a local illumination model. Typically, these maps describe the surface geometry, reflectance and illumination properties in color maps, bump maps, displacement maps, normal maps, relief maps, horizon maps, shadow maps, etc. The resolution of the mesh and of the maps is then chosen to provide enough details in close-up views. However, in many cases an object can be seen from a wide range of distances. Then, for distant views, many mesh triangles and many map *texels* of a single object can project to the same screen pixel. An accurate and anti-aliased rendering requires a weighted average over this *footprint*, *i.e.*, an integral of the contributions of all these elements, which can be extremely costly.

- *INRIA (Laboratoire Jean Kuntzmann, Université de Grenoble)*
- *CNRS (Laboratoire Jean Kuntzmann, Université de Grenoble)*

Numerous adaptive multisampling methods have been proposed to compute this integral. These methods can guarantee accurate, unbiased renderings, but their rendering time can be arbitrarily high. It is generally not compatible with real-time applications, especially with very detailed scenes. Another approach is to adapt the resolution of the model, *i.e.*, its level of details, so that only a few triangles and a few map texels project to a single screen pixel, whatever the viewing distance. These *pre-filtering* methods are much more efficient than adaptive multisampling, because the coarser mesh and map resolutions of the model can be pre-computed before rendering. But the main problem is then to find *accurate* pre-filtering methods. Indeed:

- linear pre-filtering is inaccurate. The local illumination equation is a *non-linear* function of the incident radiance and of the surface maps. Hence, evaluating this function on linearly averaged texel values is only a crude approximation of its average value over these texels.
- separate pre-filtering is inaccurate when the surface features are *correlated*. For instance, if the top and bottom parts of the surface bumps have different colors, then color and visibility are correlated (at grazing angles only the top of the surface bumps is visible). Then, combining separately pre-filtered maps is only a crude approximation of the pre-filtering of their combination.
- pre-filtering the mesh and its maps separately is inaccurate. For efficient rendering at large view distances, the coarse mesh itself must be pre-

filtered. The effects on local illumination of the removed mesh details such as curvature and bumps must then be incorporated in the maps, first as normals, then as roughness parameters. A separate filtering of the mesh and its maps simply looses these details and is therefore inaccurate.

Because of the three above issues, the general problem of pre-filtering complex surfaces is still largely unsolved. Very few methods have been proposed for the last two issues (see Section 8). However, several methods have been proposed to solve the first issue, namely the pre-filtering of complex surface maps used in a non-linear way for rendering. This paper is a survey of these methods. It is organized as follows. In Section 2 we derive the general equation that pre-filtering methods should try to approximate as accurately as possible. We also present the uncorrelation hypothesis used to split this problem into simpler ones: normal, horizon and shadow map pre-filtering. We then review the existing methods for each subproblem in Sections 3, 4, 5 and 6. In Section 7, an analysis of these methods allows us to highlight three generic tools to pre-filter maps used in non-linear functions. We conclude with a discussion of the validity of the hypotheses made by these methods, and of open issues, in Sections 8 and 9.

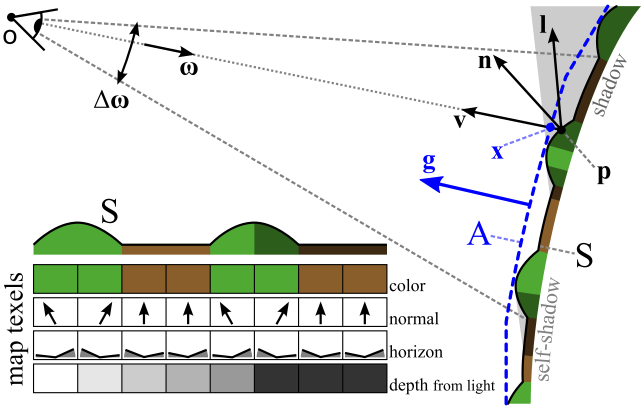


Fig. 1. **Problem statement.** S is a complex surface whose fine scale details are represented via some maps defined on a coarse mesh A (here color, normal, horizon and depth maps). We want to compute the surface radiance I in the $\Delta\omega$ cone for a pixel. This is the integral over $\Delta\omega$ of the local illumination equation, itself parameterized by the surface maps. For efficiency, we want to avoid a numerical integration over the numerous texels in $\Delta\omega$. Instead, we want to compute I from maps *pre-filtered* at a coarser resolution. The problem is that I depends non-linearly on the texel values.

2 Complex surfaces pre-filtering

The pre-filtering methods discussed in this paper aim at efficiency, but also at rendering accuracy. In order to discuss the approximations made by these methods, sometimes implicitly, we derive in Section 2.1 the general equation that pre-filtering methods are supposed to approximate as accurately as possible. We then present their main strategies to tackle this problem in Section 2.2, and the fundamental tool used to solve it, namely linear pre-filtering, in Section 2.3.

2.1 Pre-filtering equation

2.1.1 Notations

We consider a complex surface whose fine scale details are represented via some maps defined on a coarse mesh (see Fig. 1). We note S the *footprint* of a pixel on the surface, and A the orthogonal projection of S on the coarse mesh. Note that A is different from the footprint of the pixel on the coarse mesh, except for vertical views. We note this the *parallax offset* (see Fig. 7). We note $\mathbf{p}$ and $\mathbf{n}$ the points and normals of S, and $\mathbf{v}$ and $\mathbf{l}$ the unit vectors towards the viewer and a light source. We note $\mathbf{x}$ and $\mathbf{g}$ the points and normals of A. In an abuse of notations, $\mathbf{p}$ (*resp.* $\mathbf{x}$) denote either a 3D point or its 2D coordinates on the surface (*resp.* mesh), depending on the context. For instance, in $\int_S \mathrm{d}\mathbf{p}$, $\mathrm{d}\mathbf{p}$ is a 2D surface element.

A *map* is a pattern or a global function or atlas defined on the coarse mesh and storing attribute(s) of the complex surface. We note $m_\mathbf{x}$ the value of a map m at the projection of $\mathbf{x}$ along $\mathbf{g}$ on S (the surface is supposed to be a height field in the footprint S). We note m_A its average value on the projection of A along $\mathbf{g}$. For instance, $\mathbf{n}_\mathbf{x}$ is the normal at $\mathbf{x}$, and $\mathbf{n}_A$ its average value (often equal to $\mathbf{g}$). We note $m_\mathbf{x}(\ldots)$ a map describing a function at each point. For instance, $\rho_\mathbf{x}(\mathbf{l},\mathbf{v})$ is the surface BRDF at $\mathbf{x}$. In this specific case only, it is in fact a shorthand notation for a general function $\rho(\mathbf{n}_\mathbf{x},\ldots,m_\mathbf{x},\mathbf{l},\mathbf{v})$ depending on other maps defining at each point surface attributes such as the normal, the tangent vectors, the roughness, etc. We suppose that all quantities are expressed in a *constant frame*, *i.e.*, independent of $\mathbf{x}$ and $\mathbf{p}$ (like a world frame, or the tangent frame at the center of A).

We note $L(\mathbf{o},\boldsymbol{\omega})$ the radiance traveling at $\mathbf{o}$ from direction $\boldsymbol{\omega} = -\mathbf{v}$. Thus, $L(\mathbf{p},\mathbf{l})$ and $L(\mathbf{p},\boldsymbol{\omega})$ are the incident and outgoing radiance at $\mathbf{p}$, respectively. We omit the wavelength parameter λ in all radiance, color and BRDF quantities, which must then be understood as scalar quantities for a given wavelength.

We finally use two more shorthand notations: we omit, most of the time, the index of the normal map $\mathbf{n}_\mathbf{x}$, simply noted $\mathbf{n}$; and we note $\mathbf{nv} \stackrel{\text{def}}{=} \max(\mathbf{n}\cdot\mathbf{v},0)$ a scalar product clamped to 0.

2.1.2 General equation

We can now derive the general "pre-filtering equation" relating the average outgoing surface radiance in the footprint A, as a function of the surface maps used as parameters in the local illumination equation.

TABLE 1
Important symbols used in this paper.

Symbol	Description
w	low pass anti-aliasing filter (*e.g.*, a box filter)
S	footprint of a pixel on the complex surface
A	orthogonal projection of S on the coarse mesh
$\mathbf{p}$	a point on the complex surface
$\mathbf{x}$	orthogonal projection of $\mathbf{p}$ on the coarse mesh
$\mathbf{v}$	unit vector towards the viewer at $\mathbf{p}$
$\mathbf{l}$	unit vector towards a light source at $\mathbf{p}$
$\mathbf{n}$	complex surface normal at $\mathbf{p}$
$\mathbf{nv}$	clamped scalar product $\max(\mathbf{n}\cdot\mathbf{v}, 0)$
$\mathbf{g}$	coarse mesh normal at $\mathbf{x}$
$k_{\mathbf{x}}$, k_A	color at $\mathbf{x}$, average color in A
$\rho_{\mathbf{x}}$, ρ_A	BRDF at $\mathbf{x}$, average effective BRDF in A
$V_{\mathbf{x}}$, V_A	visibility at $\mathbf{x}$, average bidirectional visibility in A
$U_{\mathbf{x}}$, U_A	cast shadow at $\mathbf{x}$, average cast shadow in A
$f_{\mathbf{x}}$, f_A	function at $\mathbf{x}$, average in A
$p_{\mathbf{x}}$, p_A	probability distribution at $\mathbf{x}$, average in A

For this we start from an averaging equation in screen space and we progressively transform it to express it in terms of the maps defined on A.

For a pinhole camera with an instantaneous exposure (see Fig. 1), the intensity I of a given screen pixel is a convolution of the incident radiance $L(\mathbf{o},\boldsymbol{\omega})$ with a low-pass filter w over its angular support $\Delta\boldsymbol{\omega}$ [1], [2]. This support corresponds to a single pixel for a box filter, or to several pixels for more efficient anti-aliasing filters:

$$I = \int_{\Delta\boldsymbol{\omega}} w(\boldsymbol{\omega})L(\mathbf{o},\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \quad \text{with} \quad \int_{\Delta\boldsymbol{\omega}} w(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} = 1 \quad (1)$$

In order to get an equation in terms of surface maps, the first step is to rewrite I as an integral over the footprint S, that we define as the intersection of the surface with the $\Delta\boldsymbol{\omega}$ cone (we assume here, without loss of generality, that S fully covers this cone – if not, split it in two parts). In case of self-masking, by definition of S, a view ray intersects S several times, but only the nearest intersection point contributes to I. We exclude the others with the function $V_{\mathbf{p}}(\mathbf{v})$ equal to 1 if $\mathbf{p}$ is visible from direction $\mathbf{v}$, and 0 otherwise. The change of variables from $\boldsymbol{\omega}$ to $\mathbf{p}$ yields the Jacobian $\mathrm{d}\boldsymbol{\omega} = \frac{\mathbf{nv}}{r^2}\mathrm{d}\mathbf{p}$ where $r = \|\mathbf{p}-\mathbf{o}\|$ is the distance to the viewer. Thus (1) becomes:

$$I = \int_S w(\mathbf{p})L(\mathbf{p},\boldsymbol{\omega})V_{\mathbf{p}}(\mathbf{v})\mathbf{nv}\frac{\mathrm{d}\mathbf{p}}{r^2} \quad \text{with} \quad \int_S w(\mathbf{p})V_{\mathbf{p}}(\mathbf{v})\mathbf{nv}\frac{\mathrm{d}\mathbf{p}}{r^2} = 1 \quad (2)$$

Since the solid angle of a pixel is small r and $\mathbf{v}$ can generally be considered as constants in these integrals. Then r can be eliminated by moving it outside the integrals.

We can now use the local illumination equation $L(\mathbf{p},\boldsymbol{\omega}) = \int_\Omega L(\mathbf{p},\mathbf{l})R(\mathbf{l},\mathbf{v})\,\mathbf{nl}\,\mathrm{d}\mathbf{l}$ to express the outgoing radiance $L(\mathbf{p},\boldsymbol{\omega})$ as a function of the incident radiance $L(\mathbf{p},\mathbf{l})$ and of the surface reflectance R:

- we write the incident radiance as the product $E(\mathbf{l})V_{\mathbf{p}}(\mathbf{l})U_{\mathbf{p}}(\mathbf{l})$. $E(\mathbf{l})$ is the distant environment radiance coming from direction $\mathbf{l}$, supposed quasi constant over S, or uncorrelated with local features. $V_{\mathbf{p}}(\mathbf{l})$ is a self-shadowing term equal to 1 is $\mathbf{p}$ is visible from direction $\mathbf{l}$, and 0 otherwise. Finally, $U_{\mathbf{p}}(\mathbf{l})$ is a shadowing term equal to 1 is $\mathbf{p}$ is unoccluded by other objects in direction $\mathbf{l}$, and 0 otherwise (soft shadows result from an integral over $\mathbf{l}$). Note that $V_{\mathbf{p}}(\mathbf{l})$ excludes self inter-reflections. The local illumination equation also excludes subsurface scattering. These two effects are discussed in Section 8.
- we write the surface reflectance R as the product $k_{\mathbf{p}}\rho_{\mathbf{p}}(\mathbf{l},\mathbf{v})$, with $k_{\mathbf{p}}$ a color and $\rho_{\mathbf{p}}$ a normalized BRDF. In general a sum of such terms is used, for instance to decompose the reflectance into diffuse and specular components, but this does not impact our discussion here. Keep also in mind that both terms implicitly depend on the wavelength λ.

Putting all this together in (2), we get:

$$\begin{aligned} I &= \int_\Omega \tau_S(\mathbf{l},\mathbf{v})E(\mathbf{l})\mathrm{d}\mathbf{l} \\ \tau_S &= \frac{\int_S w(\mathbf{p})k_{\mathbf{p}}\rho_{\mathbf{p}}(\mathbf{l},\mathbf{v})U_{\mathbf{p}}(\mathbf{l})V_{\mathbf{p}}(\mathbf{l})V_{\mathbf{p}}(\mathbf{v})\mathbf{nl}\,\mathbf{nv}\mathrm{d}\mathbf{p}}{\int_S w(\mathbf{p})V_{\mathbf{p}}(\mathbf{v})\mathbf{nv}\mathrm{d}\mathbf{p}} \end{aligned} \quad (3)$$

The last step to get an equation in terms of surface maps is to rewrite I as an integral over the *coarse mesh*, on which the maps are defined. This change of variables yields the Jacobian $\mathrm{d}\mathbf{x} = \mathbf{ng}\,\mathrm{d}\mathbf{p}$, which gives:

$$\begin{aligned} I &= \int_\Omega \tau_A(\mathbf{l},\mathbf{v})E(\mathbf{l})\mathrm{d}\mathbf{l} \\ \tau_A &= \frac{\int_A w(\mathbf{x})k_{\mathbf{x}}\rho_{\mathbf{x}}(\mathbf{l},\mathbf{v})V_{\mathbf{x}}(\mathbf{l})V_{\mathbf{x}}(\mathbf{v})U_{\mathbf{x}}(\mathbf{l})\mathbf{nl}\frac{\mathbf{nv}}{\mathbf{ng}}\mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x})V_{\mathbf{x}}(\mathbf{v})\frac{\mathbf{nv}}{\mathbf{ng}}\mathrm{d}\mathbf{x}} \end{aligned} \quad (4)$$

This is the "pre-filtering equation" that we will consider in the rest of this paper. It gives I as a function of the texel values of the color, normal, BRDF, visibility, shadow and environment maps $k_{\mathbf{x}}$, $\mathbf{n}_{\mathbf{x}}$, $\rho_{\mathbf{x}}$, $V_{\mathbf{x}}$, $U_{\mathbf{x}}$ and $E(\mathbf{l})$ – with $V_{\mathbf{x}}$ often represented with an horizon map (see Section 4). Note that the parallax offset is implicit in this equation. Indeed, A is *not* the footprint of the pixel on the coarse mesh, but is related to it via a view-dependent offset (see Section 8.3 and Fig. 7).

The goal of the pre-filtering methods is to avoid a numerical evaluation of (4) during rendering, over all the texels $\mathbf{x}$. Instead, the goal is to estimate I directly from *pre-filtered* versions of the surface maps. The problem is that (4) gives I as a non-linear and non-separable function of the map texels. Thus, a simple separate linear pre-filtering of each map is not sufficient.

Note that I includes a spatial integral over A, and a directional integral over $\mathbf{l}$. Thus, both the spatial maps and the environment map $E(\mathbf{l})$ should be pre-filtered for efficient rendering. In this survey we restrict ourselves to the pre-filtering of the spatial maps, *i.e.,* to the computation of τ_A. A survey of the environment map pre-filtering methods can be found in [3].

2.2 Pre-filtering strategies

A first pre-filtering strategy is to store precomputed or measured values of $\tau_A(\mathbf{l},\mathbf{v})$ in (4), for a large number of view and light directions, and for many footprints A of different sizes covering the coarse mesh. The values can then be stored in a 6D table called a Bidirectional Texture Function, or BTF [4]. Since this approach is directly based on the general pre-filtering equation, it automatically takes into account the parallax offset and the correlation effects that are modeled by this equation. It can also take self inter-reflections and subsurface scattering effects into account, although they were neglected in (4). The main drawback of this approach is that it requires a lot of memory: an uncompressed BTF represents gigabytes of data. Several methods have been proposed to acquire a BTF from real measurements [4], to create a BTF from Monte-Carlo simulations on a geometric surface model [5], and to compress BTFs into manageable representations, in particular for interactive rendering [5], [6], [7]. A survey of these methods can be found in [8]. We do not discuss them here, nor the closely related image-based rendering methods assuming a fixed light or view direction such as [9], [10]. They are out of our scope because they do not propose algorithms to pre-filter maps used in a non-linear way for rendering.

Another pre-filtering strategy is to assume that the surface colors, normals, visibility and shadows are uncorrelated, which is sometimes really the case (we discuss the validity of this hypothesis in Section 8). This hypothesis allows one to replace the τ_A term in (4) with a product of simpler terms. Indeed, if two variables a and b are uncorrelated in the sense that $\int \mathrm{w}(a-\overline{a})(b-\overline{b}) = 0$, with $\bar{a} = \frac{\int \mathrm{w}a}{\int \mathrm{w}}$ and $\bar{b} = \frac{\int \mathrm{w}b}{\int \mathrm{w}}$, then we have $\overline{ab} = \overline{a}\,\overline{b}$. Applying this to the numerator in (4) with the weight $\mathrm{w} = w\frac{\mathbf{nv}}{\mathbf{ng}}$ we get:

$$I \approx \int_\Omega k_A \rho_A(\mathbf{v},\mathbf{l}) V_A(\mathbf{v},\mathbf{l}) U_A(\mathbf{l}) E(\mathbf{l}) \mathrm{d}\mathbf{l} \quad (5)$$

$$k_A \stackrel{\text{def}}{=} \frac{\int_A w(\mathbf{x}) k_\mathbf{x} \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}} \quad (6)$$

$$\rho_A(\mathbf{l},\mathbf{v}) \stackrel{\text{def}}{=} \frac{\int_A w(\mathbf{x}) \rho_\mathbf{x}(\mathbf{l},\mathbf{v})\, \mathbf{nl} \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}} \quad (7)$$

$$V_A(\mathbf{l},\mathbf{v}) \stackrel{\text{def}}{=} \frac{\int_A w(\mathbf{x}) V_\mathbf{x}(\mathbf{l}) V_\mathbf{x}(\mathbf{v}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x}) V_\mathbf{x}(\mathbf{v}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}} \quad (8)$$

$$U_A(\mathbf{l},\mathbf{v}) \stackrel{\text{def}}{=} \frac{\int_A w(\mathbf{x}) U_\mathbf{x}(\mathbf{l}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x}) \frac{\mathbf{nv}}{\mathbf{ng}} \mathrm{d}\mathbf{x}} \quad (9)$$

In practice most methods use the following hypothesis, most often implicitly. We discuss its validity in Section 8.

Hypothesis 1: The surface maps are uncorrelated, the parallax offset relating A to S can be ignored, as well as the *parallax Jacobian* $\mathbf{nv}/\mathbf{ng}$.

We present in Sections 3, 4 and 5, respectively, the pre-filtering methods to compute the average reflectance ρ_A, the average visibility V_A and the average shadow U_A. Before that, we now recall the basic linear pre-filtering methods that can be used with linear parameters such as the surface color k_A.

2.3 Linear pre-filtering

2.3.1 Hypotheses

The linear terms in the pre-filtering equation, such as the color map $k_\mathbf{x}$, can be linearly pre-filtered if:

- the parallax offset, implicit in (6), is either null or can be handled separately – see Section 8.
- the parallax Jacobian $\mathbf{nv}/\,\mathbf{ng}$ is quasi constant or uncorrelated with $k_\mathbf{x}$, and thus cancels when moved outside both integrals in the fraction (6).

If these two hypotheses hold, the average color becomes a simple weighted average of the color map $k_\mathbf{x}$ over A:

$$k_A \approx \int_A \hat{w}(\mathbf{x}) k_\mathbf{x} \mathrm{d}\mathbf{x} \quad \text{with } \hat{w}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{w(\mathbf{x})}{\int_A w(\mathbf{x})\mathrm{d}\mathbf{x}} \quad (10)$$

Note that when the coarse mesh curvature is small ($\mathbf{g} \approx cste$) the above hypotheses hold if the view is nearly vertical ($\mathbf{v} \approx \mathbf{g}$), or if the surface bumps are small ($\mathbf{n} \approx \mathbf{g}$).

2.3.2 Algorithms

The problem of pre-filtering a texture $k_\mathbf{x}$ to efficiently evaluate its weighted average k_A in the footprint A with (10) has been extensively studied.

The main problem is to approximate an almost arbitrary footprint A with a finite number of predefined footprints. A common solution is to use nested footprints. MIP-maps [11] use predefined nested footprints given by a quadtree. Summed Area Tables [12], or SAT, use all the rectangles whose bottom left corner is the one of the texture. They are more flexible than MIP-maps but require floating point numbers (unlike MIP-maps they do not store bounded averages, but potentially unbounded *sums* – as the name implies). Also their accuracy is orientation-dependent, yielding annoying sharpness oscillations when the surface rotates. Both methods compute k_A with a small and constant number of arithmetic operations. But they are limited to a box filter w, with axis aligned square or rectangle footprints. However, in general, a footprint is trapezoidal or elliptical [1], not axis-aligned, and w is a more complex filter less prone to aliasing than the box filter. In this case, k_A can be approximated by

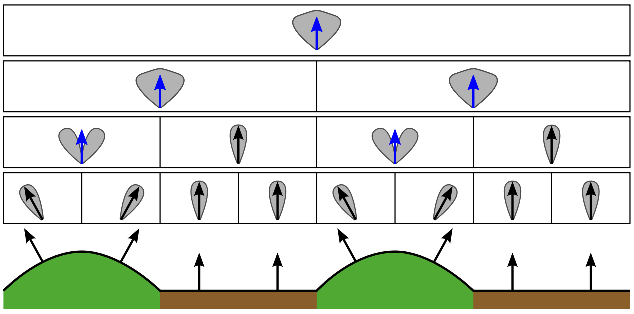


Fig. 2. **Normal map pre-filtering.** Pre-filtering the normals *(black arrows)* of a complex surface *(bottom)* quickly gives an unnormalized average vertical normal *(blue)* that does not accurately represent the surface (*e.g.*, seen from space an ocean looks flat but does not reflect light like a mirror). Instead, the solution is to pre-filter normal distribution functions, or NDFs *(gray lobes)*. The problem is then to find compact, accurate and linearly interpolable NDF representations.

using a (weighted) decomposition of A into a small number of square footprints [13], [14], [15].

Some methods are now efficiently implemented in hardware [14], [16]. It is therefore tempting to take advantage of this hardware acceleration, even for maps used in a non-linear way for rendering. This is one of the reasons why the methods to pre-filter these maps generally try to reformulate them in terms of new parameters that *can* be linearly filtered.

3 Normal Map Pre-Filtering

The average reflectance $\rho_A(\mathbf{l},\mathbf{v})$ in (7) can be computed from a *normal map*. A normal map [17] is a texture that stores a normal per texel. The main effect of a normal map $\mathbf{n_x}$ on the pre-filtering equation (4) is to perturb the diffuse and specular reflections, via its influence on the BRDF – we recall here that $\rho_{\mathbf{x}}(\mathbf{l},\mathbf{v})$ is a shorthand for $\rho(\mathbf{n_x},\ldots,\mathbf{l},\mathbf{v})$.

A normal map cannot be directly pre-filtered: the average reflectance in a footprint A is *not* equal to the reflectance due the average normal, because the BRDF is non-linear. As shown below, a solution is to pre-filter instead the probability distribution of the normals, called a *normal distribution function*, or NDF. The problem is then to find compact, accurate and linearly interpolable NDF representations. Before presenting the methods that have been proposed for that, we present the hypothesis made by all these methods, sometimes implicitly.

3.1 Hypotheses

The normal map pre-filtering methods [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], presented below, start from Hypothesis 1. Then (7) becomes:

$$\rho_A(\mathbf{l},\mathbf{v}) \approx \int_A \hat{w}(\mathbf{x})\rho_{\mathbf{x}}(\mathbf{l},\mathbf{v})\mathbf{nl}\,\mathrm{d}\mathbf{x} \tag{11}$$

Then, the methods cited above assume that the 4D BRDF $\rho_{\mathbf{x}}(\mathbf{l},\mathbf{v})$ has some symmetries allowing it to be represented with a 2D function $f_{\mathbf{x}}(\mathbf{h})$ of the half-vector $\mathbf{h}$ between $\mathbf{l}$ and $\mathbf{v}$. More precisely, they assume, most often implicitly, that $\rho_{\mathbf{x}}$ has the following separable form [25]:

$$\rho_{\mathbf{x}}(\mathbf{l},\mathbf{v}) = r(\mathbf{l},\mathbf{v})\frac{f_{\mathbf{x}}(\boldsymbol{\nu}(\mathbf{l},\mathbf{v}))}{\mathbf{nl}} \tag{12}$$

where

- $r(\mathbf{l},\mathbf{v})$ is a function *independent of* $\mathbf{x}$ *and* $\mathbf{n}$, which can represent, for instance, a Fresnel reflectance coefficient $R+(1-R)(1-\mathbf{vh})^5$ [28];
- $\boldsymbol{\nu}(\mathbf{l},\mathbf{v})$ is a function *independent of* $\mathbf{x}$ *and* $\mathbf{n}$, which generally represents the half-vector $\mathbf{h}$ (this is not mandatory);
- $f_{\mathbf{x}}(\boldsymbol{\eta})$ is a distribution function of *micro-scale normals*, such as microfacet normals in microfacet-based BRDF models. This micro-scale NDF, or μNDF, *depends on* $\mathbf{x}$, at least via its normal: we have $f_{\mathbf{x}}(\boldsymbol{\eta}) = f(\mathbf{n_x},\ldots,\boldsymbol{\eta})$, with optional maps defining at each point the tangent vectors, the surface roughness, etc.

This general form is true, for instance, for the Lambertian BRDF $\rho_L = 1$ and for the Blinn-Phong [29] lobe $\rho_{BP} = \frac{(\mathbf{nh})^\alpha}{\mathbf{nl}}$. However, this is not true for many BRDF models (see Section 3.3). The methods cited above can now be divided into *direct* and *convolution* methods.

3.1.1 *Direct methods*

The advantage of the hypothesis used in (12) is that the BRDF $\rho_{\mathbf{x}}$, which was a non-linear function of the normal map $\mathbf{n_x}$, is now given as a *linear* function of the micro-scale normal distribution function $f_{\mathbf{x}}(\boldsymbol{\eta})$. The μNDF can thus be linearly pre-filtered:

$$f_A(\boldsymbol{\eta}) \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x})f_{\mathbf{x}}(\boldsymbol{\eta})\mathrm{d}\mathbf{x} \tag{13}$$

and the average reflectance ρ_A in (11) simply becomes $\rho_A(\mathbf{l},\mathbf{v}) \approx r(\mathbf{l},\mathbf{v})f_A(\mathbf{h})$. The methods used to pre-filter these μNDF are presented in Section 3.2.

3.1.2 *Convolution methods*

The convolution methods [23], [25] use the additional hypothesis that the μNDF $f_{\mathbf{x}}$ only depends on $\mathbf{n}$, *i.e.*, that $f_{\mathbf{x}}(\boldsymbol{\eta}) = f(\mathbf{n},\boldsymbol{\eta})$. This allows them to replace the integral over A in (13) with an integral using the distribution function of the normals $\mathbf{n}$ in A. This NDF, noted $p_A(\boldsymbol{\eta})$, is defined by the ergodicity equation:

$$\forall\varphi,\ \int_A \hat{w}(\mathbf{x})\varphi(\mathbf{n_x})\mathrm{d}\mathbf{x} = \int_\Omega \varphi(\boldsymbol{\eta})p_A(\boldsymbol{\eta})\mathrm{d}\boldsymbol{\eta} \tag{14}$$

Reporting (15) below in (14) shows that the NDF in a footprint A is a weighted average of the NDFs at each point $\mathbf{x}$, which are Dirac distributions:

$$p_A(\boldsymbol{\eta}) = \int_A \hat{w}(\mathbf{x})p_{\mathbf{x}}(\boldsymbol{\eta})\mathrm{d}\mathbf{x},\quad p_{\mathbf{x}}(\boldsymbol{\eta}) \stackrel{\text{def}}{=} \delta(\boldsymbol{\eta}-\mathbf{n_x}) \tag{15}$$

Then, by using (14) in (13) the μNDF in a footprint A becomes the *convolution* of the punctual μNDF f with the NDF p_A in A:

$$f_A(\boldsymbol{\eta}') = \int_\Omega f(\boldsymbol{\eta}, \boldsymbol{\eta}') p_A(\boldsymbol{\eta}) \mathrm{d}\boldsymbol{\eta} \tag{16}$$

The pre-filtering problem is thus decomposed in two sub-problems: pre-filtering the NDFs p_A, and computing the above convolution. In practice the convolution methods [23], [25] assume that $f_{\mathbf{x}}(\boldsymbol{\eta}) = f(\mathbf{n}\boldsymbol{\eta})$ – we have $f_L = \mathbf{n}\boldsymbol{\eta}$ and $f_{BP} = (\mathbf{n}\boldsymbol{\eta})^\alpha$ for the Lambert and Blinn-Phong models, respectively. This restricts them to uniform, isotropic BRDFs (while the direct methods support anisotropic BRDFs varying with $\mathbf{x}$).

3.1.3 Summary

The convolution methods pre-filter the NDF of the macroscopic surface normals and convolve it with a punctual μNDF to get the average μNDF in a footprint. The direct methods pre-filter this average μNDF directly. Both use symmetry hypotheses to represent a 4D BRDF with a 2D μNDF. In both cases, as said above, the pre-filtering problem is reduced to the pre-filtering of NDFs. The problem is then to find compact, accurate and linearly interpolable NDF representations. We now present the existing solutions to this problem.

3.2 Algorithms

The normal map filtering algorithms can be separated in two groups: those using a single lobe to represent a NDF, and those using multiple lobes.

3.2.1 Single lobe algorithms

Schilling [20] models the μNDF f_A with a Gaussian. He represents it with the 2×2 covariance matrix of the micro-scale *slopes*, expressed in a tangent frame aligned with $\mathbf{n}$, the average micro-scale normal. Because each μNDF is expressed in its own local tangent frame, this representation *cannot* be linearly MIP-mapped and interpolated.

Olano and North [21] model the μNDF f_A with a 3D Gaussian defined by its mean $\boldsymbol{\mu}$ and its 3×3 symmetric covariance matrix Σ:

$$f_A(\boldsymbol{\eta}) \propto \exp\left(-\frac{1}{2}(\boldsymbol{\eta} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{\eta} - \boldsymbol{\mu})\right) \tag{17}$$

The advantage of this representation is that the best 3D Gaussian fit of a mixture of 3D Gaussians is obtained by linear interpolation. More precisely the mean $\boldsymbol{\mu}$ and the *second moments* $\Sigma + \boldsymbol{\mu}\boldsymbol{\mu}^T$ of the best Gaussian fit are linear interpolations of the means and second moments of the input Gaussians (see Section 7). Hence, usual MIP-mapping, bilinear and trilinear interpolation methods can be used on these 9 parameters, *before* computing lighting with (17). This method is thus very efficient. It also supports anisotropic NDFs. The drawback is that a single Gaussian cannot represent accurately μNDFs with clearly distinct lobes.

Neyret [22] models the μNDF f_A with the distribution of the normals to an ellipsoid (his method is designed for volumes but could be used as well for surfaces). For an ellipsoid defined by the points $\mathbf{p}$ such that $\mathbf{p}^T Q \mathbf{p} = 1$, this distribution function is:

$$f_A(\boldsymbol{\eta}) = \det Q^{-1} (\boldsymbol{\eta}^T Q^{-1} \boldsymbol{\eta})^{-2} \tag{18}$$

As with Gaussians, a mixture of such μNDFs is not a μNDF of the same form, but it can be approximated with such a μNDF, with a matrix Q^{-1} given by the linear interpolation of the Q^{-1} matrices of the input μNDF. Hence, these matrices can be linearly MIP-mapped and interpolated. Since they are symmetric and can be multiplied by a constant without changing f_A, this method requires only 5 parameters per texel.

Olano and Baker [27] model the μNDF f_A with a single Gaussian, as in Olano and North [21]. However, here they use a 2D Gaussian in a tangent plane above the surface:

$$f_A(\boldsymbol{\eta}) \propto \exp\left(-\frac{1}{2}(\tilde{\boldsymbol{\eta}} - \tilde{\boldsymbol{\mu}})^T \Sigma^{-1} (\tilde{\boldsymbol{\eta}} - \tilde{\boldsymbol{\mu}})\right) \tag{19}$$

where $\tilde{\boldsymbol{\eta}} = [\eta_x \ \eta_y]^T / \eta_z$. Since a common plane is used for all the μNDFs, their first and second moments $\tilde{\boldsymbol{\mu}}$ and $\Sigma + \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^T$ can be linearly MIP-mapped and interpolated. Here this gives only 5 parameters per texel, because the Gaussians are 2D.

Toksvig [23] uses a convolution method (see Section 3.1.2). He models the *macroscopic* NDF p_A with an isotropic Gaussian of the angular deviation from the average normal. This requires 4 parameters for the average normal $\boldsymbol{\mu}$ and for the variance σ^2. However, Toksvig only stores $\boldsymbol{\mu}$, computed with a linear MIP-mapping of the normal map. He then uses the norm of this average normal to estimate the variance σ^2: if the norm is almost 1 this means that all the normals $\mathbf{n}$ are almost aligned, and so that σ^2 is almost 0. Conversely, if the norm is less than 1, then σ^2 is not null. In practice, Toksvig uses $\sigma^2 \approx (1 - \|\boldsymbol{\mu}\|)/\|\boldsymbol{\mu}\|$. Then, in order to compute the convolution in (16), he also approximates f, assumed here to be the Blinn-Phong lobe $(\mathbf{nh})^\alpha$, with a Gaussian. This gives the convolution of two Gaussians, which is itself a Gaussian. The result is a Blinn-Phong lobe with a scaled down exponent α', *i.e.*, a wider lobe [23]:

$$f_A(\boldsymbol{\eta}) = \frac{1 + \alpha'}{1 + \alpha} \left(\frac{\boldsymbol{\mu}\boldsymbol{\eta}}{\|\boldsymbol{\mu}\|}\right)^{\alpha'} \quad \text{with } \alpha' \stackrel{\text{def}}{=} \frac{\alpha}{1 + \alpha\sigma^2}$$

3.2.2 Multiple lobes algorithms

Fournier [18], [19] represents the μNDF f_A with a sum of Blinn-Phong [29] lobes of the form:

$$f_A(\boldsymbol{\eta}) = \sum_k a_k (\boldsymbol{\mu}_k \boldsymbol{\eta})^{\alpha_k} \tag{20}$$

He uses up to 7 lobes per texel, which gives 28 parameters per texel. Each level of the MIP-map pyramid is not computed from the previous level, but directly from the base level, using a non-linear least square method to find the lobes that best fit the base level NDFs. The a_k, $\boldsymbol{\mu}_k$ and α_k parameters can not be linearly interpolated. Thus, for rendering, Fournier interpolates the *lighting* due to the neighboring texels. This means evaluating (20) with 28 terms, or 56 terms with trilinear filtering. This representation is quite accurate but not very adapted for real-time rendering.

Tan et al. [24], [26] extend this approach. Instead of Blinn-Phong lobes, they use a mixture of isotropic Gaussian lobes to represent the μNDF:

$$f_A(\boldsymbol{\eta}) = \sum_k a_k \exp\left(-\|\boldsymbol{\eta} - \boldsymbol{\mu}_k\|_{xy}^2 / \sigma_k^2\right) \tag{21}$$

They also improve the non-linear optimization phase to ensure that the lobe directions $\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}'_k$ in any two neighboring texels are as close as possible, for each index k. Thanks to this "alignment" they can use hardware bilinear or trilinear filtering to interpolate the means $\boldsymbol{\mu}_k$ and second moments $\sigma_k^2 + \|\boldsymbol{\mu}_k\|^2$ of corresponding lobes, before using (21) to compute the lighting. Tan et al. use 4 lobes per texel, which gives 12 parameters per texel. A drawback of this method is that it is not always possible to align the lobes in neighboring texels, for instance at sharp edges.

Han et al. [25] use a convolution method (see Section 3.1.2). They model the *macroscopic* NDF p_A with its decomposition into spherical harmonics. This gives p_{lm} coefficients that can be linearly MIP-mapped and interpolated. Then, the spherical harmonics coefficients of the μNDF f_A are simply the spherical harmonics coefficients of p_A, times the zonal harmonics coefficients f_l of the μNDF $f_{\mathbf{x}}$. This gives:

$$f_A(\boldsymbol{\eta}) = \sum_{l,m} f_l p_{lm} Y_{lm}(\boldsymbol{\eta})$$

Han et al. use up to 64 p_{lm} coefficients per texel. For a Blinn-Phong exponent α, about $4(\alpha + \sqrt{\alpha})$ coefficients are needed. This limits this method to cases where the μNDFs and the NDFs are low frequency (*i.e.,* no specular reflections, and NDF without narrow lobes).

In the other cases, Han et al. [25] model the macroscopic NDF p_A with a mixture of von Mises-Fisher (vMF) lobes:

$$p_A(\boldsymbol{\eta}) = \sum_k a_k \frac{\kappa_k \exp(\kappa_k\, \boldsymbol{\mu}_k \boldsymbol{\eta}\,)}{4\pi \sinh \kappa_k} \tag{22}$$

Like Tan et al. [24], they propose a non-linear optimization method to find the lobes corresponding to a NDF, and to align corresponding lobes in neighboring texels (which is not always possible, as said above). They can then interpolate their coefficients (κ_k and $\boldsymbol{\mu}_k$ do not interpolate linearly but a_k and $a_k(\coth \kappa_k - \kappa_k^{-1})\boldsymbol{\mu}_{\boldsymbol{k}}$ do). The result is:

$$f_A(\boldsymbol{\eta}) \approx \sum_{k,l} a_k f_l \sqrt{\frac{2l+1}{4\pi}} \exp\left(-\frac{l^2}{2\kappa_k}\right) Y_{l0}(\boldsymbol{\mu}_k \boldsymbol{\eta})$$

3.3 Discussion

Each algorithm is well adapted, under some hypotheses, to a specific type of surface. Those using a single symmetric lobe are well adapted, for instance, to smooth isotropic surfaces (*i.e.,* derivable and rotation invariant surfaces). Those using a single asymmetric lobe are well adapted to smooth anisotropic surfaces, such as those made of micro-cylinders. Those using multiple symmetric lobes are well adapted to surfaces with sharp edges, resulting in a small number of privileged normal directions, often found with manufactured surfaces. Methods using multiple asymmetric lobes would be well adapted to a very large class of surfaces, at the cost of higher memory needs.

However, all these algorithms share the same limitations. Some are coming from the hypothesis that correlations and parallax offsets or Jacobians can be neglected. They are discussed in Section 8. The main other limitation comes from the symmetry hypotheses used to be able to represent a 4D BRDF with a 2D μNDF, as in (12). Indeed, many realistic BRDFs do not have this symmetry. For instance the geometrical attenuation factor of the Cook-Torrance model [30], or the normalization factor of the Ward model [31], accounting for self-shadowing and self-masking effects at the micro-scale, *cannot* be reduced via symmetries to 2D functions. They cannot be included in the $r(\mathbf{l}, \mathbf{v})$ term in (12) either, because they depend on $\mathbf{n}$. Hence, the above algorithms cannot be accurate for grazing view or light angles, where self-masking and self-shadowing become important (even if they were correctly taken into account at the macro-scale A).

4 Horizon map pre-filtering

The average visibility $V_A(\mathbf{l}, \mathbf{v})$ in (8), *i.e.,* the proportion of texels in the footprint A that are not masked or shadowed by S itself, can be computed from an *horizon map*. An horizon map [32] is a texture containing in each texel the elevation angle of the horizon for each azimuthal direction. A point on the surface is visible from a distant light if the light angle is greater than the horizon angle, and hidden otherwise (see Fig. 3). This can be used to compute self-shadows on a complex surface (see Fig. 1).

An horizon map cannot be directly pre-filtered: the average number of visible texels in a footprint cannot

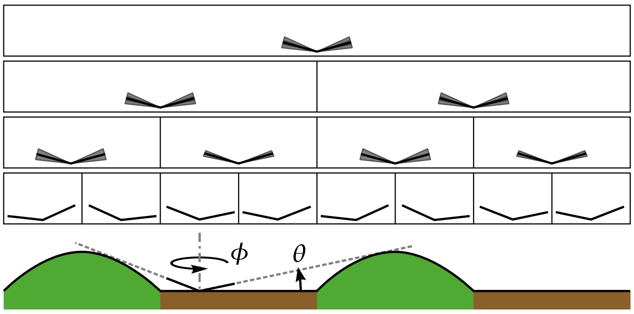


Fig. 3. **Horizon map pre-filtering**. An horizon map contains horizon elevation angles θ *(black lines)* for each azimuthal direction ϕ *(figure plane)*. It can be used to compute the remote visibility of points on a complex surface *(bottom)*. Since visibility is a non-linear function of these angles, an horizon map cannot be linearly interpolated or pre-filtered. A solution is to pre-filter horizon angle distributions, or HDF *(gray lobes)*. The problem is then to find compact, accurate and linearly interpolable HDF representations.

be computed from their average horizon angles, because the visibility function is a non-linear function of these angles. As shown below, a solution is to pre-filter instead the probability distribution of the horizon angles, called a *horizon distribution function*, or HDF. The problem is then to find compact, accurate and linearly interpolable HDF representations. Before presenting the methods that have been proposed for that, we present the hypothesis made by all these methods, sometimes implicitly.

4.1 Hypotheses

The horizon map pre-filtering methods [18], [26] start from Hypothesis 1. Then the average visibility V_A, given by (8), becomes:

$$V_A(\mathbf{l},\mathbf{v}) \approx \frac{\int_A w(\mathbf{x}) V_{\mathbf{x}}(\mathbf{l}) V_{\mathbf{x}}(\mathbf{v}) \mathrm{d}\mathbf{x}}{\int_A w(\mathbf{x}) V_{\mathbf{x}}(\mathbf{v}) \mathrm{d}\mathbf{x}} \tag{23}$$

The visibility function $V_{\mathbf{x}}$ is initially represented with an *horizon map* $\Theta_{\mathbf{x}}(\phi)$ (see Fig. 3):

$$V_{\mathbf{x}}(\mathbf{v}) = H(\theta_v - \Theta_{\mathbf{x}}(\phi_v)) \tag{24}$$

where $\Theta_{\mathbf{x}}(\phi)$ is the horizon angle in azimuth ϕ, and H is the Heaviside function. Several strategies are used to model the azimuthal dependence of $\Theta_{\mathbf{x}}$. For instance, using a fixed set of azimuthal directions $\phi_1, \ldots, \phi_n$, with a piecewise linear interpolation [18], [26], or a B-spline interpolation [33]. It is also possible to fit the 2D curve $(\cos\Theta_{\mathbf{x}}(\phi), \phi)$ in polar coordinates with an ellipse [34].

As said above, because the Heaviside function in (24) is non-linear, the horizon map $\Theta_{\mathbf{x}}(\phi)$ cannot be linearly pre-filtered. To solve this problem, Fournier [18] and Tan et al. [26] reformulate this equation by using an *horizon distribution function*, or HDF, noted $p_{\mathbf{x}}(\theta,\phi)$ and equal to $\delta(\theta - \Theta_{\mathbf{x}}(\phi))$:

$$V_{\mathbf{x}}(\mathbf{v}) = \int_0^{\frac{\pi}{2}} H(\theta_v - \theta) p_{\mathbf{x}}(\theta, \phi_v) \mathrm{d}\theta \tag{25}$$

The advantage of their reformulation is that the visibility function $V_{\mathbf{x}}$, which was a non-linear function of the horizon map $\Theta_{\mathbf{x}}$, is now a *linear* function of the HDF $p_{\mathbf{x}}$. These distributions can thus be linearly pre-filtered, using $p_A(\theta,\phi) \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) p_{\mathbf{x}}(\theta,\phi) \mathrm{d}\mathbf{x}$, in order to compute the proportion of texels that are visible in the footprint A, $V_A(\mathbf{v}) \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) V_{\mathbf{x}}(\mathbf{v}) \mathrm{d}\mathbf{x}$:

$$\begin{aligned} V_A(\mathbf{v}) &= \int_0^{\frac{\pi}{2}} H(\theta_v - \theta) p_A(\theta, \phi_v) \mathrm{d}\theta \\ &= \int_0^{\theta_v} p_A(\theta, \phi_v) \mathrm{d}\theta \end{aligned} \tag{26}$$

This gives a method to pre-filter the denominator of (23). The same principles can be used for the numerator, but then we get a 4D function $p_A(\theta,\phi,\theta',\phi')$ equal to $\int_A \hat{w}(\mathbf{x}) p_{\mathbf{x}}(\theta,\phi) p_{\mathbf{x}}(\theta',\phi') \mathrm{d}\mathbf{x}$. Encoding a 4D function in each texel is very costly, so Fournier [18] and Tan et al. [26] approximate (23) with an equation of the form:

$$V_A(\mathbf{l},\mathbf{v}) \approx \mathcal{V}(V_A(\mathbf{l}), V_A(\mathbf{v}), \mathbf{l}, \mathbf{v}) \tag{27}$$

This allows them to use only 1D HDF distributions $p_A(\theta,\phi_i)$, $i = 1 \ldots n$, while still being able to take into account some correlation effects between $\mathbf{l}$ and $\mathbf{v}$ in (23), discussed below. The problem is then to find compact, accurate and linearly interpolable HDF representations. We now present the existing solutions to this problem.

4.2 Algorithms

Fournier [18] represents the HDF $p_A(\theta,\phi_i)$ for each azimuth ϕ_i with its mean $\mu_A(\phi_i)$ and standard deviation $\sigma_A(\phi_i)$. As seen in Section 3, these parameters can be linearly MIP-mapped and interpolated, by storing the second moment instead of the standard deviation. Fournier then uses these pre-filtered parameters to approximate (26) with a linear ramp going from $V_A = 0$ for $\theta_v = \mu_A - \sigma_A$ to $V_A = 1$ for $\theta_v = \mu_A + \sigma_A$:

$$V_A(\mathbf{v}) \approx \operatorname{clamp}\left(\frac{\theta_v - \mu_A(\phi_v) + \sigma_A(\phi_v)}{2\sigma_A(\phi_v)}, 0, 1\right)$$

He then uses an uncorrelation hypothesis to approximate the proportion of texels in the footprint A that are both visible and illuminated, with the product of the visible and illuminated texels proportions:

$$V_A(\mathbf{l},\mathbf{v}) \approx V_A(\mathbf{l}) V_A(\mathbf{v})$$

In practice Fournier uses 4 ϕ_i directions aligned with the texture axis, which gives 8 parameters per texel (4 means and 4 second moments).

To address some correlations between normals and visibility (see Section 8), Fournier proposes to use one

HDF per Blinn-Phong lobe (see Section 3). Then, for pre-filtered, correlated normal and visibility maps, 84 parameters per texel are needed (28 for the 7 lobes, plus 56 for the 7 occlusion maps)!

Tan et al. [26] also represent the HDF $p_A(\theta, \phi_i)$ for each azimuth ϕ_i with its mean $\mu_A(\phi_i)$ and standard deviation $\sigma_A(\phi_i)$. This gives $2n$ parameters per texel for n ϕ directions (Tan et al. do not give the value used in their examples). However, they use more accurate approximations than Fournier to compute (26) and (23) from these pre-filtered parameters. First, they assume that the HDFs p_A are Gaussians. Then (26) can be computed exactly with the erf function. Second, they approximate (23) with:

$$V_A(\mathbf{l},\mathbf{v}) \approx \frac{1}{V_A(\mathbf{v})}\Big[\alpha \min\big(V_A(\mathbf{l}), V_A(\mathbf{v})\big) + (1-\alpha)\max\big(V_A(\mathbf{l}) + V_A(\mathbf{v}) - 1, 0\big)\Big] \quad (28)$$

where $\alpha = (1 + \mathbf{lv})/2$. The $\min$ term corresponds to the maximum possible fraction of A that can be visible from both $\mathbf{l}$ and $\mathbf{v}$. The $\max$ term corresponds to the minimum possible fraction. The α term interpolates between the two to account for greater correlations when $\mathbf{l}$ and $\mathbf{v}$ are close (when $\mathbf{l} = \mathbf{v}$, $V_A(\mathbf{l})$ and $V_A(\mathbf{v})$ are fully correlated). In particular, this approximation correctly reproduces the *hotspot effect*, which is the fact that when $\mathbf{l}$ and $\mathbf{v}$ are aligned all the self-shadows are masked to the viewer, and then all the visible areas are lit. Mathematically this means that $V_A(\mathbf{l},\mathbf{v})$ should be equal to 1 when $\mathbf{l} = \mathbf{v}$. This is easily checked on (23) and on (28), but this is not the case with Fournier's method. This approximation also correctly reproduces the fact that if the area A is fully lit, *i.e.*, if $V_A(\mathbf{l}) = 1$, then $V_A(\mathbf{l},\mathbf{v})$ must be equal to 1. Again, this is easily checked on (23) and on (28), but this is not the case with Fournier's approximation.

It should be noted here that Ashikhmin et al. [35], in the context of BRDF models, propose a similar approximation:

$$V_A(\mathbf{l},\mathbf{v}) \approx \frac{\beta \min\big(V_A(\mathbf{l}), V_A(\mathbf{v})\big) + (1-\beta)V_A(\mathbf{l})V_A(\mathbf{v})}{V_A(\mathbf{v})}$$

where β is a Gaussian of the angle between $\mathbf{l}$ and $\mathbf{v}$. It is easy to check that this method also gives 1 when $\mathbf{l} = \mathbf{v}$ (hotspot effect), and when the area is fully lit. All above approximations correctly give $V_A(\mathbf{l},\mathbf{v}) = V_A(\mathbf{l})$ when A is fully visible, *i.e.*, when $V_A(\mathbf{v}) = 1$.

Finally, we should also mention that among all the papers that use precomputed visibility functions on height fields, some use other representations that can be linearly MIP-mapped and interpolated to compute an average visibility over an arbitrary area (although this was not the primary goal of the authors). This is the case for instance of Nowrouzezahrai and Snyder [33]. Indeed, they decompose the Heaviside function in (24) on the linear basis of the normalized Legendre polynomials $\hat{P}_k$, which gives:

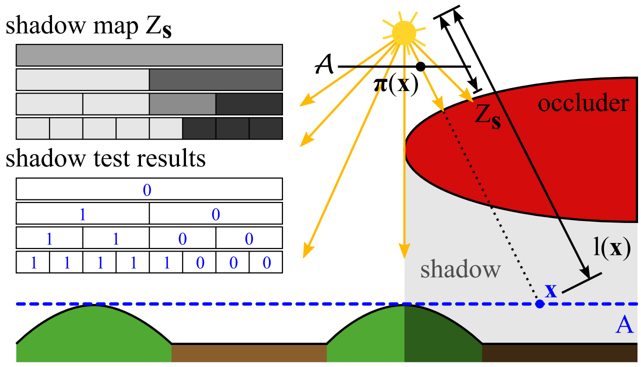


Fig. 4. **Shadow map pre-filtering.** A shadow map is a depth map Z_s *(gray texels)* of the scene from the point of view of a light source. A point $\mathbf{x}$ is lit if its depth $l(\mathbf{x})$ is less than or equal to $Z_{\boldsymbol{\pi}(\mathbf{x})}$, where $\boldsymbol{\pi}(\mathbf{x})$ is the projection of $\mathbf{x}$ in the shadow map. Performing this binary depth test *(blue numbers)* on the coarsest linearly pre-filtered depth value gives 0 here, instead of the correct result $5/8$ (the average of the shadow test over the 8 base texels) because the depth test is non linear. Shadow map pre-filtering methods aim at solving this problem.

$$V_{\mathbf{x}}(\mathbf{v}) = \sum_k a_k(\Theta_{\mathbf{x}}(\phi_v))\hat{P}_k(\cos\theta_v) \quad (29)$$

The advantage of this decomposition is that the visibility function $V_{\mathbf{x}}$, which was a non-linear function of the horizon map $\Theta_{\mathbf{x}}$, is now a *linear* function of the basis coefficients $a_k(\Theta_{\mathbf{x}})$. These coefficients can thus be linearly pre-filtered to compute an average visibility:

$$V_A(\mathbf{v}) = \sum_k \left(\int_A \hat{w}(\mathbf{x})a_k(\Theta_{\mathbf{x}}(\phi_v))\mathrm{d}\mathbf{x}\right)\hat{P}_k(\cos\theta_v) \quad (30)$$

5 SHADOW MAP PRE-FILTERING

The average visibility $U_A(\mathbf{l},\mathbf{v})$ in (9), *i.e.*, the proportion of texels in the footprint A that are not shadowed by objects other than S, can be computed with many methods: shadow volumes [37], shadow maps [38], deep shadow maps [39], opacity shadow maps [40], etc. A survey of these methods can be found in [41] and [42]. Here we consider only the case of shadow maps [38], for which non-linear pre-filtering methods are needed, and have been proposed. A shadow map is a *depth map* of the scene, computed from the point of view of a punctual light source. Then, a binary *depth test* suffice to compute shadows: a point $\mathbf{x}$ is lit by this source if and only if its depth is less than or equal to the depth stored in the shadow map (see Fig. 4).

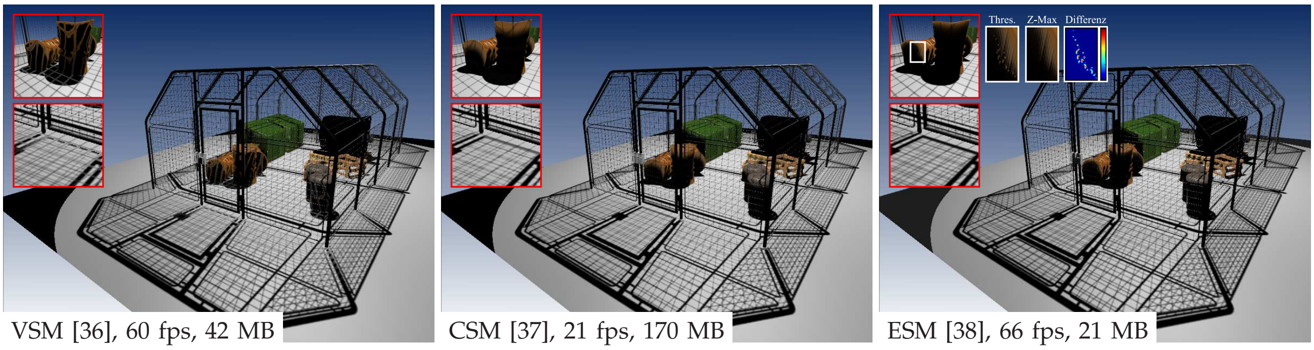


Fig. 5. **Shadow map pre-filtering algorithms.** A complex scene rendered with a 2k×2k shadow map (from [36], courtesy of Thomas Annen and Max-Planck Institute for Informatics). Exponential Shadow Maps (*ESM*) are the most efficient, with better contact shadows than Convolution Shadow Maps (lower closeups) and less light leaking than Variance Shadow Maps (top closeups).

A shadow map cannot be directly pre-filtered: the average number of lit texels in a footprint A cannot be computed from their average depth, because the depth test is a non-linear function of these depths. As shown below, one solution is to pre-filter instead the probability distribution of these depths, that we call here a *depth distribution function*, or ZDF. The problem is then to find compact, accurate and linearly interpolable ZDF representations. Before presenting the methods that have been proposed for that, we present the hypothesis made by all these methods, sometimes implicitly.

5.1 Hypotheses

The shadow map pre-filtering methods [36], [43], [44], [45], presented below, start from Hypothesis 1. Then the average visibility U_A, given by (9), becomes:

$$U_A \approx \int_A \hat{w}(\mathbf{x}) U_{\mathbf{x}} \mathrm{d}\mathbf{x} \tag{31}$$

where we dropped $\mathbf{l}$ (for a punctual light source $\mathbf{l}$ only depends on $\mathbf{x}$, and so $U_{\mathbf{x}}(\mathbf{l}(\mathbf{x}))$ can be noted $U_{\mathbf{x}}$). When using shadow maps, $U_{\mathbf{x}}$ is not stored explicitly, and so cannot be directly pre-filtered with (31). Instead, it is represented with a binary depth test applied to a *depth map* $Z_{\mathbf{s}}$. More precisely, by noting $l(\mathbf{x})$ and $\boldsymbol{\pi}(\mathbf{x})$ the depth of $\mathbf{x}$ and its projection in the shadow map, respectively (see Fig. 4), $U_{\mathbf{x}}$ is computed with:

$$U_{\mathbf{x}} = \mathcal{U}_{\boldsymbol{\pi}(\mathbf{x})}(l(\mathbf{x})) \quad \text{with } \mathcal{U}_{\mathbf{s}}(d) \stackrel{\text{def}}{=} H\big(Z_{\mathbf{s}} - d\big) \tag{32}$$

and (31) becomes $U_A \approx \int_{\mathcal{A}} \hat{w}(\mathbf{s}) H\big(Z_{\mathbf{s}} - l(\mathbf{x})\big) \mathrm{d}\mathbf{s}$, where $\mathcal{A}$ is the projection of the footprint A in the shadow map. The $l(\mathbf{x})$ term in this equation prevents any pre-filtering of the shadow map alone. To solve this problem the shadow map pre-filtering methods assume that $l(\mathbf{x})$ can be approximated with its average l, which gives:

$$U_A \approx \mathcal{U}_{\mathcal{A}}(l) \quad \mathcal{U}_{\mathcal{A}}(d) \stackrel{\text{def}}{=} \int_{\mathcal{A}} \hat{w}(\mathbf{s}) H\big(Z_{\mathbf{s}} - d\big) \mathrm{d}\mathbf{s} \tag{33}$$

Still, as said above, because the Heaviside function in (32) and (33) is non-linear, the depth map $Z_{\mathbf{s}}$ cannot be *linearly* pre-filtered. To solve this, one solution is to use *depth distribution functions*, or ZDF. We present this method and other solutions below.

5.2 Algorithms

Donnely and Lauritzen [43] reformulate (32) by using the ZDF $p_{\mathbf{s}}(z) \stackrel{\text{def}}{=} \delta(z - Z_{\mathbf{s}})$:

$$\mathcal{U}_{\mathbf{s}}(d) = \int_0^\infty H(z-d) p_{\mathbf{s}}(z) \mathrm{d}z \tag{34}$$

The advantage of their reformulation is that the visibility function $\mathcal{U}_{\mathbf{s}}$, which was a non-linear function of the depth map $Z_{\mathbf{s}}$, is now a *linear* function of the ZDF $p_{\mathbf{s}}$. These distributions can thus be linearly pre-filtered, with $p_{\mathcal{A}}(z) \stackrel{\text{def}}{=} \int_{\mathcal{A}} \hat{w}(\mathbf{s}) p_{\mathbf{s}}(z) \mathrm{d}\mathbf{s}$, in order to compute the proportion of texels that are lit in A:

$$\begin{aligned} \mathcal{U}_{\mathcal{A}}(d) &= \int_0^\infty H(z-d) p_{\mathcal{A}}(z) \mathrm{d}z \\ &= \int_d^\infty p_{\mathcal{A}}(z) \mathrm{d}z \end{aligned} \tag{35}$$

Then, Donnely and Lauritzen represent the ZDF $p_{\mathcal{A}}(z)$ with its first and second moments $\mu_{\mathcal{A}}$ and $\sigma_{\mathcal{A}}^2 + \mu_{\mathcal{A}}^2$, which can be linearly MIP-mapped and interpolated, as we have already seen in previous sections. Finally, they use Chebychev's inequality to approximate $\mathcal{U}_{\mathcal{A}}$ with its upper bound (this choice gives the exact result for a planar occluder parallel to the surface):

$$\mathcal{U}_{\mathcal{A}}(d) \approx \mathcal{U}_{\mathcal{A}}^{\max}(d) = \frac{\sigma_{\mathcal{A}}^2}{\sigma_{\mathcal{A}}^2 + (d - \mu_{\mathcal{A}})^2} \text{ if } d > \mu_{\mathcal{A}}, \text{ else } 1$$

This gives anti-aliased hard shadows with only two values per shadow map texel. The main drawback is that light bleeding (*i.e.*, wrongly lit areas inside shadows) can occur when the variance $\sigma_{\mathcal{A}}^2$ is high.

Annen et al. [44] decompose the Heaviside function $H(z-d)$ in (32) by using a basis of $B_k(z)$ functions.

This can be seen as a decomposition, for each d, of a function $H_d(z) \stackrel{\text{def}}{=} H(z-d)$. Then the basis coefficients depend on d, which gives:

$$H(Z_\mathbf{s} - d) = \sum_k a_k(d) B_k(Z_\mathbf{s}) \quad (36)$$

The advantage of their decomposition is that the visibility function $\mathcal{U}_\mathbf{s}$, which was a non-linear function of the depth map $Z_\mathbf{s}$, is now a *linear* function of the $B_k(Z_\mathbf{s})$ coefficients. These coefficients can thus be linearly pre-filtered:

$$\mathcal{U}_\mathcal{A}(d) = \sum_k a_k(d) \left(\int_\mathcal{A} \hat{w}(\mathbf{s}) B_k(Z_\mathbf{s}) \mathrm{d}\mathbf{s} \right) \quad (37)$$

In practice Annen et al. decompose the Heaviside function with its Fourier series. This gives an expression with the form expected in (36):

$$\begin{aligned} H(Z_\mathbf{s} - d) &= \sum_k c_k \exp\left(\frac{2i\pi k}{T}(Z_\mathbf{s} - d) \right) \\ &= \sum_k c_k \exp\left(-\frac{2i\pi k}{T} d \right) \exp\left(\frac{2i\pi k}{T} Z_\mathbf{s} \right) \end{aligned}$$

Annen et al. use 16 coefficients, which gives 16 parameters per shadow map texel. This is not sufficient to avoid artifacts due to ringing (Gibbs phenomenon), but they propose scaling and offsetting methods to reduce these artifacts. This method can be extended to support soft shadows [46].

In another paper [36], Annen et al. reuse the idea of decomposing H in separable terms of d and z. Instead of using a Fourier series, they simply approximate H with an exponential (Salvi [47] uses the same idea). Indeed, for $c \gg 1$, $\exp(cx)$ has a very steep slope near $x = 0$ and quickly tends towards 0 for $x < 0$, like the step function H. This gives:

$$H(Z_\mathbf{s} - d) \approx \exp(c(Z_\mathbf{s} - d)) = \exp(-cd) \exp(cZ_\mathbf{s})$$

Here again, the advantage of this approximation is that the visibility function $\mathcal{U}_\mathbf{s}$ becomes a *linear* function of $\exp(cZ_\mathbf{s})$. These warped depths can thus be linearly pre-filtered, unlike the depth map $Z_\mathbf{s}$ itself:

$$\mathcal{U}_\mathcal{A}(d) \approx \exp(-cd) \left(\int_\mathcal{A} \hat{w}(\mathbf{s}) \exp(cZ_\mathbf{s}) \mathrm{d}\mathbf{s} \right) \quad (38)$$

This gives anti-aliased hard shadows with only one parameter per shadow map texel. The problem of this approximation is that it diverges for $d < Z_\mathbf{s}$. Annen et al. propose some solutions to fix this. Lauritzen and McCool [45] propose to combine this exponential warping with its negative counterpart $-\exp(-cZ_\mathbf{s})$, together with Chebychev's inequality [43]. This solution avoids the divergence problem, but requires 4 parameters per shadow map texel – first and second moments for the warped ZDFs $\exp(cZ_\mathbf{s})$ and $-\exp(-cZ_\mathbf{s})$.

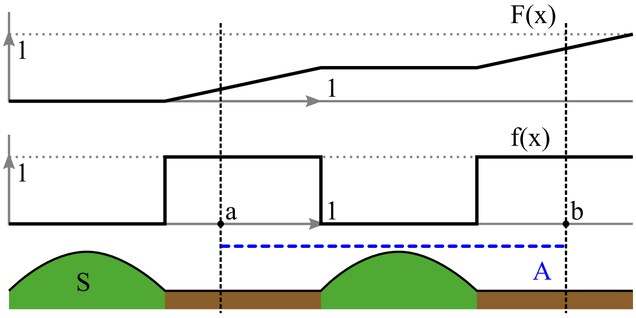


Fig. 6. **Procedural map filtering.** The color of the complex surface S *(bottom)* can be described procedurally with the analytical function $f(x) = \lfloor 2x \rfloor \div 2$ *(middle)*, using $f\,\mathrm{brown} + (1-f)\,\mathrm{green}$. The average color in the footprint A can be computed efficiently by using the indefinite integral $F(x) = \frac{1}{2}\lfloor x \rfloor + \max(x - \lfloor x \rfloor - \frac{1}{2}, 0)$ *(top)*, with $\int_a^b f(x)\mathrm{d}x = F(b) - F(a)$ (assuming $\hat{w}$ is a box filter).

6 Procedural Map Filtering

In the previous sections the maps were stored explicitly in textures. It is also possible to use procedural maps evaluated on the fly during rendering. These maps cannot be pre-filtered before rendering, otherwise they would no longer be procedural. Still, since they have an explicit analytical form, it is possible to approximate their low-pass filtering analytically to get an efficient anti-aliased rendering, without using multisampling. In this section we review the methods that have been proposed for that, for the case of procedural color and normal maps (we are not aware of any procedural horizon map or procedural shadow map filtering methods).

6.1 Procedural color map filtering

Three main classes of methods have been proposed for the case of procedural maps whose values are used linearly to compute final pixel colors. They are explained below, through the example of procedural color maps. The first two methods can be used with arbitrary procedural functions, called *shaders* [48], using conditional statements, loops, function calls, arbitrary mathematical functions, and even textures. The third method is specific to shaders using *spectral synthesis* [49], [50].

6.1.1 *Global fading*

This method progressively blends the procedural color with a constant color equal to its average [49], [50]. The blending starts when the size of the low-pass filter $\hat{w}$ becomes larger than the smallest features of the procedural texture (*i.e.,* when the apparent size of these features becomes smaller than a pixel). This method can be used either on the output of the whole shader, or only on some parts (a function, a sub expression in a function, etc). However, this method is not very accurate nor very effective: it attenuates *all* the frequencies at once, instead of filtering only the frequencies above the filter's cutoff frequency. On the other hand, this method can be automated thanks to its simplicity [51], [52].

6.1.2 Analytic integration

A much more accurate and more effective method is to use the indefinite integral F of the color function f [49], [50]. Then a filtered version of this function with a box filter can be efficiently computed, with $\int_a^b f(x)\mathrm{d}x = F(b) - F(a)$ (see Figure 6). The problem is that it is almost always impossible to find the indefinite integral F. Another problem is that the box filter is not ideal.

For instance, even a simple 2D grid pattern, which can be described with the product $f(x)f(y)$, with f a pulsetrain function $f(x) = \lfloor 2x \rfloor \div 2$ (see Figure 6), is really hard to filter *exactly* with this method, even with a box filter (due to perspective projection, the filter support becomes an arbitrary quadrilateral in the xy space). In these cases, a simple but often very effective approximation is to apply the method on each term of the procedural function (this is an approximation because these terms are generally combined non-linearly in the final result). In the above example, the 2D filtering integral is reduced to a product of two 1D integrals which are easy to compute (see Figure 6). A frequent use of this method is to replace step functions with filtered versions to anti-alias sharp edges (the built-in GLSL `smoothstep` function is done for that).

6.1.3 Frequency clamping

The third method is specific to the case where the color function f is an explicit sum of terms with a band limited spectrum. This is the case for instance when summing sinusoids, or with Perlin noise [53] (which is not strictly band limited, but other noise functions are [54]). In this case an approximate filtered function can be obtained by clamping the sum to the terms whose frequency is less than the low-pass filter cutoff frequency, and by progressively attenuating the others to avoid popping [49], [50], [55].

This method works well with a basic sum of terms $f(\mathbf{x}) = \sum_i b_i(\mathbf{x})$. In many cases however, f is used in a color lookup table c to get the final pixel color. When c is not linear, the average final color $\int_A \hat{w}(\mathbf{x})c(f(\mathbf{x}))\mathrm{d}\mathbf{x}$ is not equal to $c(\int_A \hat{w}(\mathbf{x})f(\mathbf{x})\mathrm{d}\mathbf{x})$. To solve this problem, Hart et al. [56] use the following approximation:

$$\int_A \hat{w}(\mathbf{x})c(f(\mathbf{x}))\mathrm{d}\mathbf{x} \approx \frac{1}{f_1 - f_0}\int_{f_0}^{f_1} c(\alpha)\mathrm{d}\alpha \tag{39}$$

where f_0 and f_1 are two samples of f on the boundary of A (this corresponds to the hypothesis that f varies linearly between f_0 and f_1 in A). The right hand side is then evaluated with a pre-filtered lookup table, either using MIP-maps or SATs. Worley [57] proposes an extension to this method using more samples of f to get more accurate results. Bergner et al. [58] use a spectral analysis to tackle this problem.

More generally, it is frequent to compose f, or each of its term b_i, with arbitrary functions g or g_i. For instance, Musgrave [59] uses $f(\mathbf{g}(\mathbf{x}))$ to distort a noise pattern f. Perlin [60] uses $g(\mathbf{f}(\mathbf{x}))$ to distort a density field g, and $f(\mathbf{x}) = \sum_i |b_i(\mathbf{x})|$ to get a turbulence pattern. These extensions often add high frequencies to f, and the frequency clamping method must be adapted to account for this. However, we are not aware of any generic algorithm for doing that.

6.2 Procedural normal map filtering

In most cases the procedural normal maps, used for instance in water shaders, are simply filtered with the methods of Section 6.1 to get an average normal, which is then used directly in the BRDF for final rendering. This is the case for instance of Hinsinger et al. [61]. This strategy removes aliasing but gives a wrong illumination, because the BRDF is not linear (see Section 3).

In order to solve this problem, van Horn et al. [62] propose an automatic method to convert a procedural shader, applied to a given object, into what they call reduction maps. These maps are ordinary textures, which can be MIP-mapped, containing at each point a compact description of the surface BRDF and a distribution of normals (represented with Olano's method [21] – see Section 3). These BRDFs and NDFs are computed with a stochastic sampling of the procedural shader, during a preprocessing pass. However, van Horn et al. only use a normal distribution to decide if its average normal can be used directly for rendering (*i.e.,* if the distribution is very narrow). If not, they resort to multiple sampling of the original procedural shader to filter it.

Bruneton et al. [63] propose a specific solution for the ocean case. Their idea is to compute not only an average normal, but also a NDF, in order to compute the BRDF corresponding to the details filtered out from the procedural normal map. They model the ocean surface with an explicit sum of trochoids with well defined frequencies. Then they compute an anti-aliased average normal by clamping this sum to the frequencies smaller than the cut-off frequency of the low-pass filter, as in [61]. In addition, they use the clamped frequencies to compute the variance of the distribution of normals corresponding to these frequencies. They finally use this variance to control the size of the lobe of the BRDF model used for final rendering. In this way, no details are lost due to the filtering, and a correct lighting is obtained at the end.

7 Generic tools

All the methods presented in Sections 3, 4 and 5 seem to use quite similar tools to pre-filter maps used in non-linear functions. We show here that three generic tools can be extracted from these methods, and we discuss their advantages and drawbacks. We start with a formal definition of the pre-filtering problem for non-linear functions.

7.1 Problem statement

We consider a function f of some spatially varying parameters $\mathbf{m}$, whose values are defined in a map $\mathbf{M}_\mathbf{x}$. Each texel of this map can contain several attributes, such as a color and a normal, which may be correlated. The function f may also depend on other, *non* spatially varying parameters, denoted by the vector $\mathbf{y}$. By definition, these additional parameters are uncorrelated with the map values. For instance, $\mathbf{y}$ can represent view and light directions. As a concrete example, for uncorrelated normal maps, horizon maps and shadow maps we have, respectively:

$$\begin{array}{lll} f = \rho(\mathbf{n_x}, \boldsymbol{\eta}) & \mathbf{M_x} = \mathbf{n_x} & \mathbf{y} = \boldsymbol{\eta} = \boldsymbol{\nu}(\mathbf{l}, \mathbf{v}) \\ f = H(\theta_v - \Theta_\mathbf{x}) & \mathbf{M_x} = \Theta_\mathbf{x} & \mathbf{y} = \theta_v \\ f = H(Z_\mathbf{s} - d) & \mathbf{M_s} = Z_\mathbf{s} & \mathbf{y} = d \end{array}$$

We suppose that f is a non-linear function of its spatially varying parameters, such that the average $f_A(\mathbf{y}) \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) f(\mathbf{M_x}, \mathbf{y}) \mathrm{d}\mathbf{x}$ is not equal to $f(\mathbf{M}_A, \mathbf{y})$, the value of f on the averaged parameters, defined by $\mathbf{M}_A \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) \mathbf{M_x} \mathrm{d}\mathbf{x}$. We also suppose that we cannot or do not want to store f in a precomputed table. For instance, if $\mathbf{y}$ has n components, an $n+2$ dimensional table is required to store f, which may exceed the available memory. Instead, we want to compute the average value of f in a footprint A from a pre-filtered map. The problem is that the $\mathbf{M_x}$ map cannot be linearly pre-filtered, because f is a non-linear function of its $\mathbf{m}$ parameters.

7.1.1 Direct methods

As can be seen from Sections 3, 4 and 5, a general solution to the above problem is to reformulate f as a linear function, *i.e.*, to use a "change of variables" such that f becomes a *linear* function of the new variables. Indeed, if $f(\mathbf{M_x}, \mathbf{y})$ can be rewritten as a linear function $g(\mathbf{K_x}, \mathbf{y})$ of some new parameters $\mathbf{k}$, then the average value of f in a footprint A can be computed from the linearly pre-filtered $\mathbf{K_x}$ map:

$$f_A(\mathbf{y}) = g(\mathbf{K}_A, \mathbf{y}), \text{ with } \mathbf{K}_A \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) \mathbf{K_x} \mathrm{d}\mathbf{x} \qquad (40)$$

The problem is then to find a linear reformulation of f. For this, note that the relation $f_A(\mathbf{y}) = g(\mathbf{K}_A, \mathbf{y})$ above indicates that $\mathbf{K}_A$ should somehow characterize the function f_A (because these two terms are the only ones depending on A). In other words, $\mathbf{K}_A$ is a function of f_A: $\mathbf{K}_A = \mathcal{K}(f_A)$. Then the linearity of g implies that $\mathcal{K}$ is linear: $\mathcal{K}\left(\int_A \hat{w}(\mathbf{x}) f_\mathbf{x} \mathrm{d}\mathbf{x}\right) = \int_A \hat{w}(\mathbf{x}) \mathcal{K}(f_\mathbf{x}) \mathrm{d}\mathbf{x}$. This means that, in order to find a linear reformulation of f, it is necessary (but not sufficient) to find linear parameters characterizing the functions f_A. We present three generic methods for doing that in the next sections.

7.1.2 Convolution methods

A generic method to reformulate f as a linear function is to introduce the probability distribution of its $\mathbf{m}$ parameters, that we call here a *parameter distribution function*, or PDF, noted $p_\mathbf{x}(\mathbf{m}) = \delta(\mathbf{m} - \mathbf{M_x})$:

$$f(\mathbf{M_x}, \mathbf{y}) = g(p_\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \int_M f(\mathbf{m}, \mathbf{y}) p_\mathbf{x}(\mathbf{m}) \mathrm{d}\mathbf{m} \qquad (41)$$

where M is the range of $\mathbf{m}$ (Ω for normals, $[0, \frac{\pi}{2}]$ for horizon angles, $[0, \infty[$ for depths, etc). The reformulated function g, which can be seen as a *convolution*, is now a linear function of the PDF. These distributions can thus be linearly pre-filtered, with $p_A(\mathbf{m}) \stackrel{\text{def}}{=} \int_A \hat{w}(\mathbf{x}) p_\mathbf{x}(\mathbf{m}) \mathrm{d}\mathbf{x}$, to compute the average of f in a footprint:

$$f_A(\mathbf{y}) = g(p_A, \mathbf{y}) = \int_M f(\mathbf{m}, \mathbf{y}) p_A(\mathbf{m}) \mathrm{d}\mathbf{m} \qquad (42)$$

The only problem is that the PDF is a function: it cannot be stored directly in a map like a scalar or a vector. The solution is to store linear parameters characterizing this distribution, using the same tools as for the direct methods above, which are presented in the next sections.

7.1.3 Summary

In order to pre-filter a map used in a non-linear function f the generic solution is to use a *different* map containing new, linear parameters. These parameters should characterize either the average function, or the distribution of its parameters. The first method gives the average function directly, while the second one requires the computation of a convolution (this is more complex but f can then be changed at runtime).

We now review three generic tools to characterize a function with linear parameters, which have been used in the surveyed papers: using moments, linear bases, or spanning sets. Each tool can be used directly or via a convolution.

7.2 Using moments

A first method to characterize a function φ with linear parameters $\mathbf{k}$ is to use its *moments*:

$$\mathbf{k} = \left[\int \varphi(\mathbf{y}) \mathrm{d}\mathbf{y}, \int \varphi(\mathbf{y}) \mathbf{y} \mathrm{d}\mathbf{y}, \int \varphi(\mathbf{y}) \mathbf{y}\mathbf{y}^T \mathrm{d}\mathbf{y}, \ldots\right]^T$$

This can be used both with the direct and convolution methods.

7.2.1 Direct method

The direct method uses the moments of the average function $f_A(\mathbf{y})$, which can be linearly MIP-mapped in a map $\mathbf{K}_A$. The reformulated function g is then chosen such that $g(\mathbf{K}_A, \mathbf{y})$, seen as a function of $\mathbf{y}$, has moments $\mathbf{K}_A$.

This method has been used for normal maps, with 2 moments and with g Gaussian [21], [27] (see (17) and (19) in Section 3). In this case the 0^{th} moment is not needed: it is always 1 because here f_A is a probability distribution.

7.2.2 Convolution method

The convolution method uses the moments of the PDF $p_A(\mathbf{m})$, which can be linearly MIP-mapped in a map $\mathbf{K}_A$ (the 0^{th} moment is not needed: it is always 1 because p_A is a probability distribution). The reformulated function g is then obtained by replacing p_A with a function of moments $\mathbf{K}_A$ in (42).

This method has been used for normal maps [23], with one moment (the second one being deduced from the first) and g Gaussian (as shown by the similarity between (42) and (16) in Section 3). It has also been used for horizon maps [18], [26], with 2 moments (see the similitude between (42) and (26) in Section 4). Finally, it has also been used for shadow maps [43], [45], with 2 moments and with g the upper bound of (42) over all PDF of moments $\mathbf{K}_A$ (compare (42) with (35) in Section 5).

7.2.3 Discussion

In practice, all the methods approximate f_A with a Gaussian having the same first and second moments as f_A. If $\mathbf{y}$ (or $\mathbf{m}$ for the convolution method) has n components, then n first moments and $n(n+1)/2$ second moments must be stored per texel (which gives a total of 2, 5 and 9 parameters for $n \leq 3$). Alternatively, if off-diagonal elements in the second moment matrix can be neglected, then only $2n$ parameters are needed. Thus, the advantage of using moments is that it does not require a lot of memory. The drawback is that a single Gaussian lobe is insufficient to accurately represent distributions with several lobes (see Fig. 2).

7.3 Using a basis of functions

Another solution to characterize a function with linear parameters is to use the coefficients of its decomposition in a basis of functions. This can be used both with the direct and convolution methods.

7.3.1 Direct method

Seeing f as a set of functions $\varphi_{\mathbf{y}}(\mathbf{m})$ indexed by $\mathbf{y}$, the idea is to write each $\varphi_{\mathbf{y}}$ as a linear combination of some functions $B_k(\mathbf{m})$. This gives coefficients a_k depending on the index $\mathbf{y}$, and so the average of f $f_A(\mathbf{y}) = \int_A \hat{w}(\mathbf{x}) f(\mathbf{M_x}, \mathbf{y}) \mathrm{d}\mathbf{x}$ becomes:

$$f_A(\mathbf{y}) = \sum_k a_k(\mathbf{y}) \left(\int_A \hat{w}(\mathbf{x}) B_k(\mathbf{M_x}) \mathrm{d}\mathbf{x} \right) \tag{43}$$

We then get a linear characterization of f_A with the parameters $\mathbf{k} \stackrel{\text{def}}{=} [\ldots\ B_k(\mathbf{m})\ \ldots]^T$. The function g is obtained by truncating the above sum to a finite number of terms. Symmetrically, instead of using $B_k(\mathbf{m})$ functions, we can also write f as a linear combination of $B_k(\mathbf{y})$ functions:

$$f_A(\mathbf{y}) = \sum_k \left(\int_A \hat{w}(\mathbf{x}) a_k(\mathbf{M_x}) \mathrm{d}\mathbf{x} \right) B_k(\mathbf{y}) \tag{44}$$

We then get a linear characterization of f_A with the parameters $\mathbf{k} \stackrel{\text{def}}{=} [\ldots\ a_k(\mathbf{m})\ \ldots]^T$.

This method has been used for horizon maps, with Legendre polynomials [33] (compare (44) with (30) in Section 4). It has also been used for shadow maps, by using a decomposition of f in Fourier series [44] (compare (43) with (37) in Section 5).

7.3.2 Convolution method

The convolution method applies this decomposition in a basis of functions to the PDF. For instance, with (44) we get:

$$p_A(\mathbf{m}) = \sum_k \left(\int_A \hat{w}(\mathbf{x}) a_k(\mathbf{M_x}) \mathrm{d}\mathbf{x} \right) B_k(\mathbf{m}) \tag{45}$$

which can then be reported in the convolution (42) to get the average function f_A.

This is the method used by Han et al. [25] when they use as $\mathbf{k}$ parameters the spherical harmonics coefficients p_{lm} of p_A, and when they use their properties to efficiently evaluate the convolution in (42) (compare (42) with (16) in Section 3).

7.3.3 Discussion

The advantage of using a basis of functions is simplicity. The drawback is that it is impractical to use this method when f has high-frequencies, especially on GPU, due to the large number of coefficients to handle. This is the case of the Heaviside function for horizon and shadow map filtering, and this explains why Annen et al. [44] had to use scaling and offsetting tricks. This is also the case with highly specular BRDFs, and this is why Han et al. [25] propose another method than spherical harmonics for these cases.

7.4 Using a spanning set of functions

A third solution to characterize a function with linear parameters is to use a *spanning set*[1] of functions instead of a basis. Indeed, it can be noted from (43) and

1. $\{\mathbf{e}_1, \mathbf{e}_2, \ldots\}$ is a spanning set if any vector can be decomposed into a sum $x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \ldots$. If this decomposition is unique, the spanning set is a basis.

(44) that these equations remain linear even if the B_k functions do *not* form a basis. For instance, Annen et al. [36] use (43) with a *single* function $B(\mathbf{m}) = e^{cz}$ (and $a(\mathbf{y}) = e^{-cd}$ – compare (43) with (38) in Section 5). Conversely, it is possible to use an infinite set of functions $B_\mathbf{k}$, indexed by a continuous variable $\mathbf{k}$, such as the set of all Gaussian functions or "lobes" indexed by their moments. It is then possible to use a *sparse representation*:

$$\varphi(\mathbf{y}) \approx \sum_{i=1}^{n} a_i B_{\mathbf{k}_i}(\mathbf{y}), \text{ with } n \text{ small} \tag{46}$$

where the indices $\mathbf{k}_i$ must now be stored explicitly, in addition to the "basis" coefficients a_i.

7.4.1 Direct method

The direct method uses this decomposition on the average function f_A. This method has been used for normal maps, with $B_\mathbf{k}$ functions represented with Blinn-Phong lobes [18] or with isotropic Gaussian lobes [24], [26] (compare (46) with (20) and (21) in Section 3).

7.4.2 Convolution method

The convolution method uses this decomposition on the PDF p_A, which is then reported in the convolution (42) to get the average function f_A. This is the method used by Han et al. [25] when they decompose p_A with a small set of von Mises Fisher lobes, which they further decompose into zonal spherical harmonics to efficiently compute the convolution of (42) (compare (46) with (22) in Section 3).

7.4.3 Discussion

The advantage of using a spanning set of functions is that, in general, only a few terms are necessary to represent a function φ accurately, even if it has high frequencies (*e.g.*, narrow lobes). Although now the indices $\mathbf{k}_i$ must be stored in addition to the "basis" coefficients a_i, this generally gives more accurate results, for the same number of parameters, than using a basis of functions. Another advantage is that it is generally possible to get better interpolation methods (for instance to have lobes rotating between two configurations, instead of a simple crossfading yielding ghosting artifacts).

A first drawback of this approach is that a costly non-linear optimization is generally necessary to decompose a function in such a way [18], [24], [25], [26], whereas simple dot products would suffice to find its coefficients in an orthonormal basis. This is a problem to pre-filter a map in real-time (which is necessary for shadow maps, for instance).

The second drawback is that two neighbor footprints A and A' will generally use different "basis" functions $B_{\mathbf{k}_{A,i}} \neq B_{\mathbf{k}_{A',i}}$. Therefore, in general, the coefficients $a_{A,i}$ and $a_{A',i}$ and the indices $\mathbf{k}_{A,i}$ and $\mathbf{k}_{A',i}$ cannot be linearly averaged or interpolated. This is why Fournier [18] does not MIP-map nor interpolate them, but instead interpolate the values given by (46) for each texel (which is costly). Still, in specific cases this interpolation is justified. For instance, if $B_\mathbf{k}$ is a Gaussian of moments $\mathbf{k}$ [24], [26]. But this is accurate only if $\mathbf{k}_{A,i}$ and $\mathbf{k}_{A',i}$ are close, *i.e.*, if the "lobes" are aligned [24], [25], [26]. And this alignment is not always possible, for instance at sharp edges.

In summary, when high frequencies (*e.g.*, narrow lobes) are expected it is generally preferable to use a spanning set of functions. When no a priori information is available, using a basis of functions is generally the best choice.

8 Discussion

The methods presented in Sections 3, 4 and 5 used several simplifying assumptions: they assumed that the surface features were uncorrelated, they ignored inter-reflections and subsurface scattering, they neglected the parallax offset and the parallax Jacobian, the silhouettes and the curvature of the coarse mesh A, etc. In this section we discuss the validity of these hypotheses and present the few papers that try to avoid them: we address correlations, inter-reflections, parallax, curvature and mesh pre-filtering respectively in Sections 8.1 to 8.5.

8.1 Correlations

The methods in Sections 3, 4 and 5 assumed that color, reflectance, visibility and shadows were uncorrelated (see Section 2.2). We show here that correlations are in fact quite common, by using five examples, in Sections 8.1.1 to 8.1.5: how correlations can occur in small footprints, between color and reflectance, between color and visibility, between reflectance and visibility, and between view and light visibility. We also illustrate the effects of these correlations, and we present the few papers that try to address them (the BTF models and some BRDF models consider them, but they are out of our scope – the pre-filtering of maps used non-linearly for rendering).

8.1.1 Small footprints

In a footprint with n texels, the correlation between two maps $a_\mathbf{x}$ and $b_\mathbf{x}$ whose texel values are a_i and b_i is $\sum_i (a_i - \bar{a})(b_i - \bar{b})$, with $\bar{a} = \frac{\sum_i a_i}{n}$ and $\bar{b} = \frac{\sum_i b_i}{n}$. For two texels this gives $\frac{1}{2}(a_1 - a_2)(b_1 - b_2)$. This shows that the correlation is *never* 0 in this case, unless a or b is constant. Thus for small footprints two maps are very unlikely to be uncorrelated. Here a possible solution to this correlation problem is to use multi-sampling instead of pre-filtering when n is too small (the overhead will be bounded by construction).

8.1.2 Color and reflectance

Correlations between color and reflectance frequently occur, for instance, with surfaces made of several materials, each material having its own color and reflectance (*e.g.,* dirt or oxidation on a metal). The effects of these correlations are easy to see on an example. For instance, consider a complex surface S like the one in Fig. 2, with very dark bumps and very bright flat areas, each covering one half of S. For vertical view and light directions, the color $k_{\mathbf{x}}$ and the reflectance $\rho_{\mathbf{x}}(\mathbf{l}, \mathbf{v})$ at some wavelength λ could be $k_1 = \rho_1 = 0.1$ and $k_2 = \rho_2 = 0.9$ respectively for the dark and bright parts. The correct pixel intensity I is $\frac{1}{2}(k_1\rho_1+k_2\rho_2) = 0.41$, but the intensity computed with the uncorrelation hypothesis is completely different: $\frac{1}{4}(k_1 + k_2)(\rho_1 + \rho_2) = 0.25$.

This shows that these correlations can have important effects. Most of the normal map filtering methods can be extended in a straightforward way to handle them: it suffices to pre-filter the product of the NDF with each color channel, instead of the NDF alone. With RGB colors this gives three NDFs per texel instead of one, and therefore three times more parameters per texel. However, these weighted NDFs are no longer normalized probability distributions, which is a problem for the methods using this property (like Toksvig's method [23]). Also the NDF $f_{\mathbf{x}}$, now multiplied by $k_{\mathbf{x}}$, is no longer a function of $\mathbf{n}$ alone. This is a problem for the convolution methods, which use this hypothesis (see Section 3.1.2 – Han et al. [25] propose an extension to their method to support color maps, but this extension assumes that colors are uncorrelated with normals).

8.1.3 Color and visibility

Correlations between color and visibility[2] frequently occur, for instance, with rough surfaces whose hills and valleys have different colors (*e.g.,* due to weathering). The main effect of these correlations is that the average surface color becomes view and light dependent. For instance, the average color of the complex surface in Fig. 2 is the average between green and brown for a vertical view, and pure green for a grazing view (because then the brown parts are masked – see Fig. 9). Similarly, the brown part is totally self-shadowed at grazing light angle, resulting in a mostly green average color. Of course, these correlation effects only happen when self-masking or self-shadowing occur, *i.e.,* for "grazing" view or light directions. However, these "grazing" angles can be quite large (*e.g.,* $45°$ for $\wedge$-shaped bumps) and so these correlations are not uncommon.

Wu et al. [64] use characteristic points on S to handle these correlations in offline renderings, at the price of huge memory requirements (thousands of bytes per texel).

2. For RGB-A maps storing color and "visibility" (coverage) the solution is to pre-filter the product, *i.e.,* to use pre-multiplied alpha.

8.1.4 Reflectance and visibility

Correlations between reflectance and visibility frequently occur, for instance, with rough surfaces whose hills and valleys have different reflectances (*e.g.,* due to weathering). As with colors, the main effect of these correlations is that the average surface reflectance, *i.e.,* the NDF, becomes view and light dependent (many diffuse materials become specular at grazing angles). For instance, if the bumps of the complex surface in Fig. 2 have a diffuse reflectance, while the flat areas are purely specular, then the specular component disappears for grazing view angles (because then the flat areas are masked). Again, these correlation effects only happen when self-masking or self-shadowing occur, *i.e.,* for "grazing" view or light directions, but these "grazing" angles can be quite large, *e.g.,* with $\wedge$-shaped bumps.

The pre-filtering method proposed by Fournier [18], [19] can handle these correlation effects with a low precision, at the price of huge memory requirements (see Section 4). Max and Becker [65] get view dependent normal distributions by using a precomputed redistribution function, transforming the normal distribution for a vertical view into the distribution of normals in some other view direction (taking self-masking into account). Wu et al. [64] can also handle these correlations in offline renderings, at the price of huge memory requirements.

8.1.5 View and light visibility

Visibility from the light and from the viewer become totally correlated when the view and light directions are the same. In this case self-shadows are masked from the viewer, which gives a brighter *hotspot* in the part of the image where both directions are aligned.

As shown in Section 4, there are some methods to approximate the 4D bidirectional visibility $V(\mathbf{l}, \mathbf{v})$ function by using only the 2D visibility function $V(\mathbf{v})$. These methods reproduce a hotspot effect, but they remain approximations.

8.2 Inter-reflections and subsurface scattering

The methods in Sections 3, 4 and 5 ignored the inter-reflections of the incident light between the surface bumps inside the footprint area A. However, these inter-reflections can have an important effect for surfaces with a high albedo, or with $\wedge$-shaped bumps. Without them the surface appears darker than in reality. We also ignored subsurface scattering that could occur inside this footprint. This effect is important for translucent materials: it makes bumps appear smoother by reducing the contrast between parts that are directly lit or not.

We are not aware of any non-linear pre-filtering method taking these effects into account. There are several methods to simulate inter-reflections in height fields, such as [33], [34], [66]. However, these methods compute the inter-reflections at each frame, they do not pre-filter them. Inter-reflections are sometimes explicitly taken into account in analytical and numerical BRDF models, as in [67] or [68]. Here the inter-reflections *are* pre-filtered, but only for the scales where the individual bumps are no longer visible. The smaller scales, for which maps used non-linearly for rendering would be needed (such as a bump map), and which are in the scope of our survey, are not addressed. Subsurface scattering has also been represented with an extended BRDF, called a BSSRDF [69]. But here again, the fine scales for which maps used non-linearly for rendering would be needed are not addressed.

8.3 Parallax offset and Jacobian

The methods in Sections 3, 4 and 5 ignored the view-dependent parallax offset between A and S. They also ignored the parallax Jacobian $\frac{\mathbf{nv}}{\mathbf{ng}}$. We discuss the validity of these hypotheses in the next two sections.

8.3.1 Parallax offset

We argue here that the parallax offset can be handled separately from the pre-filtering problem. Indeed, pre-filtering a map to quickly compute its average value on arbitrary footprints A is independent of the method used to compute A.

More precisely, surface bumps whose apparent size is much larger than a pixel can be represented by the underlying coarse mesh on which the maps are defined. The parallax offset is then automatically taken into account during the projection of this mesh on screen. Conversely, surface bumps whose size is much smaller than the footprint area A do not give any visible offset. This leaves the case of surface bumps whose size is of the order of A. In this case the pixel footprint A_0 on the coarse mesh may not correspond to the correct footprint that must be used, *i.e.*, the orthogonal projection A on the coarse mesh of the pixel footprint S on the complex surface (see Fig. 7). However, computing A is independent of evaluating pre-filtered maps in A. Hence, the parallax offset can be ignored when pre-filtering complex surface maps, and treated separately during rendering. In fact, several methods have been proposed to evaluate this parallax offset, such as Displacement Mapping [70], Relief Mapping [9], [71] and Parallax Occlusion Mapping [72]. However, these methods only consider "punctual" footprints (they cast rays and not cones), and they do not consider the correct pre-filtering of the height maps used for tracing these rays. Still, they could be used as a first approximation to estimate S from the parallax offset and from its partial derivatives, evaluated at the center of A.

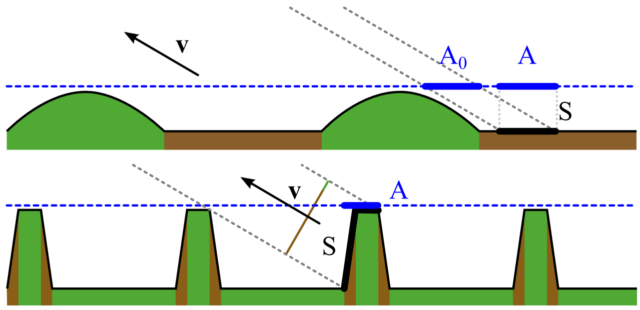


Fig. 7. **Parallax offset and Jacobian**. *Top:* for a correct rendering the surface maps must not be evaluated in A_0, the pixel footprint on the coarse mesh (*dotted blue line*), but in the orthogonal projection A of the pixel footprint S on the complex surface (here using A_0 would give a green color, instead of brown, the correct result). *Bottom:* the parallax Jacobian $\frac{\mathbf{nv}}{\mathbf{ng}}$ has important effects with near vertical facets. Here without it we find the green part as dominant, in A, while in fact, orthogonally to $\mathbf{v}$, this is the brown part.

8.3.2 Parallax Jacobian

Unlike the parallax offset, the parallax Jacobian cannot be handled separately from the pre-filtering algorithms. This Jacobian can have important effects with near vertical facets and grazing views. Indeed, an almost vertical facet has a small area in a surface map, but a large apparent area when the surface is seen at grazing angles. The Jacobian correctly accounts for this, and ignoring it can lead to errors (see Fig. 7).

As shown in Section 2.3, this Jacobian can be ignored if it is uncorrelated with the surface maps. In the other cases, recovering a correct view-dependent pre-filtered map value from a map pre-filtered without this Jacobian is not easy, because the correlation between the map and the surface normals is generally not available. Normal maps are an exception: a view-dependent NDF could be computed from a "standard" NDF by weighting each normal with $\frac{\mathbf{nv}}{\mathbf{ng}}$. However, we are not aware of any non-linear pre-filtering method doing this or, more generally, taking the parallax Jacobian into account.

8.4 Silhouettes and curvature

Silhouettes correspond, by definition, to grazing view angles. As already seen, this case is problematic for the surface map pre-filtering methods, because many correlations cannot be neglected in this situation. But the very first problem is that the coarse mesh plus surface map representation somehow breaks down in this situation, because the coarse mesh resolution becomes clearly visible at silhouettes. There are however some solutions to this problem, like Displacement Mapping [70], Silhouette Clipping [73] or Shell Mapping [74].

The curvature of the coarse mesh on which the surface maps are defined can also be a problem. For instance, a normal map with only vertical normals, once pre-filtered in texture space, will still have only vertical normals. This is clearly false for a pre-filtered area covering a curved part of the underlying coarse mesh: the normals distribution should reflect this curvature [75]. A solution is to express the normals in a global frame, instead of in the tangent space to the surface (this is possible with most normal map pre-filtering methods, except those explicitly using this tangent space, like [27]). But then the normal map cannot be a pattern mapped in a repetitive way on an object. It must also be specific for each object, and the objects cannot be deformed. Curvature is also a problem for horizon maps: the horizon angles depend on the curvature of the underlying surface. Heidrich et al. [34] pre-compute 5 horizon maps for 5 pre-defined curvatures, and then interpolate between them at runtime. However, they do not address the pre-filtering of these maps.

Note that Bidirectional Texture Functions also suffer from the silhouette and curvature problems. Wang et al. [76] solve the curvature problem in the case of a 4D "semi-BTF" where the light direction is fixed, by introducing an additional curvature parameter, yielding a 5D table. Using the same approach for a full BTF would give a 7D table, which is clearly too much (a normal BTF is already quite difficult to manage in terms of memory).

8.5 Mesh pre-filtering

The above silhouette and curvature problems are particular cases of a more general problem cited in the introduction, namely that pre-filtering the coarse mesh and its maps separately is inaccurate. Some mesh simplification algorithms take the maps into account to ensure that parallax errors introduced by the simplification step remain under a given threshold (see [77], [78] for a survey of the mesh simplification algorithms, and [79] for the specific case of terrains). This is necessary but not sufficient: the mesh details removed by the simplification must also be incorporated in the maps. Here the methods proposed by Becker et al. [65] and Schilling [20] could be inspiring, although they start from a bump map instead of a mesh, as well as the ocean filtering method of Bruneton et al. [63].

Finally, at large distances or very grazing angles, when the simplified mesh becomes very different from the complex surface, it may become impossible to represent this surface with view-independent maps (*e.g.*, the foliage of a tree cannot be defined with view-independent maps defined on an enclosing ellipsoid). At this stage, instead of using view-dependent surface maps, switching to a volumetric representation may be more appropriate. But this opens a new problem, out of the scope of this paper, namely the pre-filtering of volumetric data used in a non-linear way for rendering (*e.g.*, [80]).

9 Conclusion

Surface attributes, such as normals, can generally not be linearly pre-filtered because they are used in non-linear functions for rendering. We have presented in Sections 3 to 6 several methods addressing this problem, for the specific case of colors, normals, visibility and shadows. These methods make various restrictive assumptions, and have different quality, performance and memory requirements.

However, as discussed in Section 8, these methods are far from solving the general problem of pre-filtering the appearance of complex surfaces. Besides all the method-specific limitations, correlation of attributes is almost never accounted for, and the level of filtering is limited by the link between the coarse mesh and its attributes – which is a problem for curvature, silhouettes and distant views.

Still, we have shown in Section 7 that several generic tools for non-linear pre-filtering could be extracted from the above methods, coming from different research topics. These tools can probably be applied to more abstract or correlated parameters describing the surface appearance (*e.g.*, correlation with visibility could be treated by storing view or light-dependent parameters, using one of the representation we discussed). In fact, we believe that huge progress is feasible towards the pre-filtering of the appearance of complex surfaces in the general case.

Acknowledgment

We thank Charles De Rousiers for suggestions and proofreading.

References

[1] P. S. Heckbert, "Fundamentals of texture mapping and image warping," Berkeley, CA, USA, Tech. Rep., 1989.

[2] K. Sung, A. Pearce, and C. Wang, "Spatial-temporal antialiasing," *TVCG: IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 2, pp. 144–153, 2002.

[3] J. Kautz, P.-P. Vázquez, W. Heidrich, and H.-P. Seidel, "Unified approach to prefiltered environment maps," in *Rendering Techniques (Eurographics Workshop on Rendering'00 Proceedings)*, 2000, pp. 185–196.

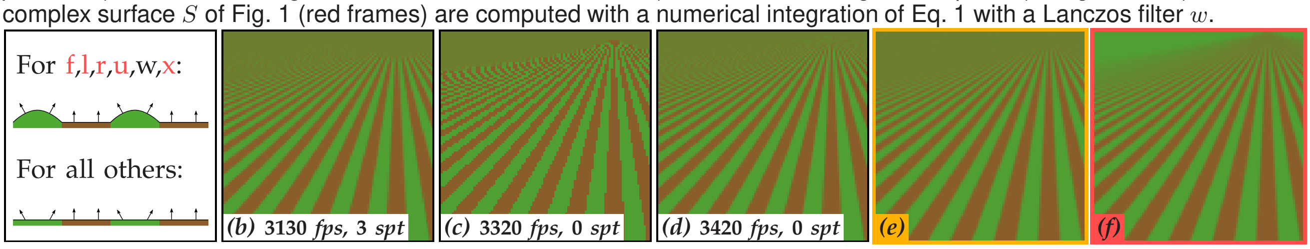
[4] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink, "Reflectance and texture of real-world surfaces," *ACM Transactions on Graphics*, vol. 18, no. 1, pp. 1–34, 1999.

[5] F. Suykens, K. V. Berge, A. Lagae, and P. Dutr, "Interactive rendering with bidirectional texture functions," *Computer Graphics Forum*, vol. 22, pp. 463–472, 2003.

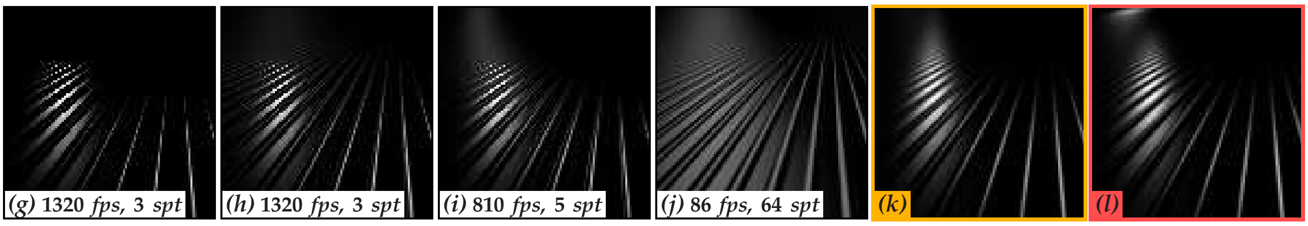
[6] W.-C. Ma, S.-H. Chao, Y.-T. Tseng, Y.-Y. Chuang, C.-F. Chang, B.-Y. Chen, and M. Ouhyoung, "Level-of-detail representation of bidirectional texture functions for real-time rendering," in *I3D'05: ACM SIGGRAPH Symposium on Interactive 3D graphics and games*. ACM, 2005, pp. 187–194.

[7] H. Wu, J. Dorsey, and H. E. Rushmeier, "A sparse parametric mixture model for btf compression, editing and rendering," *Computer Graphics Forum (Eurographics Symposium on Rendering'11 Proceedings)*, 2011.

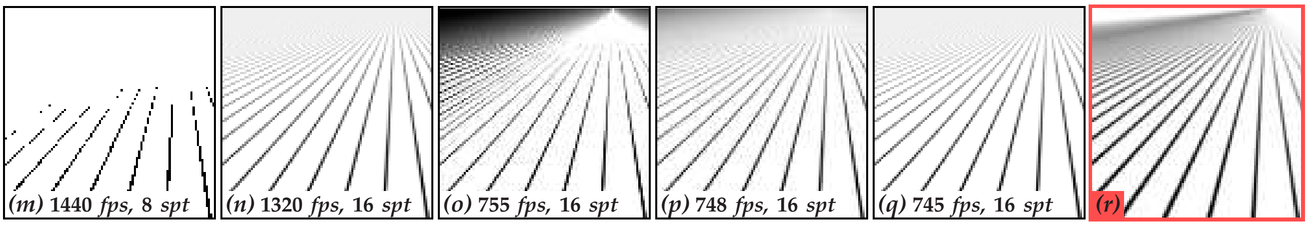
Fig. 8. **Comparisons**. Color k *(b...f)*, diffuse component $\mathbf{nl}$ *(s,t,u)*, specular component $\mathbf{nh}^{\alpha}$ *(g...l)*, self-shadows $V(\mathbf{l})$ *(m,n)*, visible self-shadows $V(\mathbf{l},\mathbf{v})$ *(o...r)* and all components together *(v,w,x)*. Performance in *frames per second* and *scalars per texel* (for 800×800 images on a NVidia GeForce GTX 470). Ground truth images for a plane (orange frames) and for the complex surface S of Fig. 1 (red frames) are computed with a numerical integration of Eq. 1 with a Lanczos filter w.



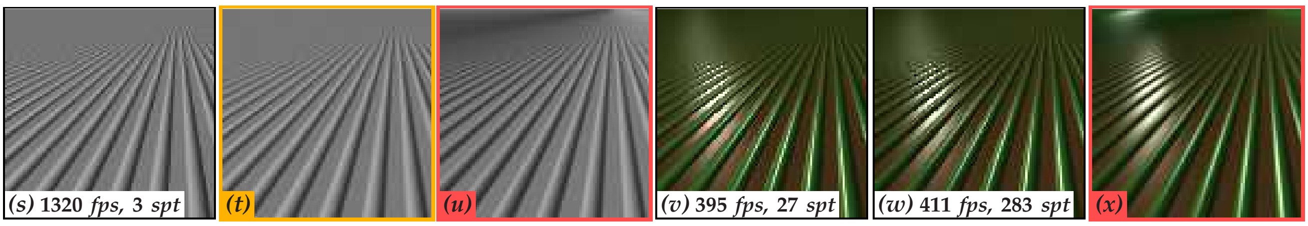
Color maps. Textures with hardware anisotropic filtering *(b)* and procedural maps with global fading *(c)* or analytic integration *(d)* are close to the reference on a plane *(e)*. However, they are quite different from the reference on surface S *(f)*, because they neglect correlations between color and visibility (in the distance, only the green bumps are visible).



Normal maps (specular). With a linearly pre-filtered normal map *(g)*, or with an isotropic lobe, *e.g.* Toksvig *(h)*, the specular highlight is lost in the distance – compare with the reference *(k)* for a plane. The result is better with an anisotropic lobe, *e.g.* Olano and Baker [27] *(i)*. Using a basis of functions, *e.g.* spherical harmonics as Han et al. [25] *(j)* does not work here, because the BRDF has high frequencies ($\alpha = 150$). Since they ignore correlations, all these methods fail to reproduce the top left highlight seen on S *(l)*, due to cumulated highlights on top of bumps (the only visible surface parts in the distance).



Horizon maps. With a linearly pre-filtered horizon map self-shadows are incorrect *(m)*. Horizon distributions [18], [26] *(n)* give better results. For *visible* self-shadows, Fournier *(o)*, Tan et al. *(p)* or Ashikhmin et al. *(q)* all give quite different results compared to the reference *(r)* for S (note the asymmetry between left and right due to different masking of shadows).



Left: **Normal maps (diffuse)**. For a bump mapped plane, a linearly pre-filtered normal map can be used to approximate the diffuse component $\int \max(\mathbf{n} \cdot \mathbf{l}, 0)$ *(t)* with $\max(\int \mathbf{n} \cdot \mathbf{l}, 0)$ *(s)*. However, this method fails to reproduce the correlation effects seen on the real surface *(u)*. *Right:* **Combined maps**. Finally, combining *(b)*, *(i)*, *(p)* and *(s)* with Eq. 5, with *(w)* or without *(v)* view-dependent displacement mapping [76], looks quite different from the ground truth *(x)*.

[8] J. Filip and M. Haindl, "Bidirectional texture function modeling: A state of the art survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1921–1940, 2009.

[9] M. M. Oliveira, G. Bishop, and D. McAllister, "Relief texture mapping," in *SIGGRAPH'00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 359–368.

[10] T. Malzbender, D. Gelb, and H. Wolters, "Polynomial texture maps," in *SIGGRAPH'01*. ACM Press/Addison-Wesley Publishing Co., 2001, pp. 519–528.

[11] L. Williams, "Pyramidal parametrics," *Computer Graphics (SIGGRAPH'83 Proceedings)*, vol. 17, no. 3, pp. 1–11, 1983.

[12] F. C. Crow, "Summed-area tables for texture mapping," in *Computer Graphics (SIGGRAPH'84 Proceedings)*, vol. 18, 1984, pp. 207–212.

[13] A. Fournier and E. Fiume, "Constant-time filtering with space-variant kernels," in *Computer Graphics (SIGGRAPH'88 Proceedings)*. ACM Press/Addison-Wesley Publishing Co., 1988, pp. 229–238.

[14] A. Schilling, G. Knittel, and W. Strasser, "Texram: A smart memory for texturing," *IEEE Computer Graphics and Applications*, vol. 16, no. 3, pp. 32–41, 1996.

[15] J. McCormack, R. Perry, K. I. Farkas, and N. P. Jouppi, "Feline: fast elliptical lines for anisotropic texture mapping," in *SIGGRAPH'99*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 243–250.

[16] M. Bóo and M. Amor, "High-performance architecture for anisotropic filtering," *Journal of Systems Architecture*, vol. 51, no. 5, pp. 297–314, 2005.

[17] J. Cohen, M. Olano, and D. Manocha, "Appearance-preserving simplification," in *SIGGRAPH'98*. ACM Press/Addison-

Wesley Publishing Co., 1998, pp. 115–122.

[18] A. Fournier, "Filtering normal maps and creating multiple surfaces," Tech. Rep., 1992.

[19] ——, "Normal distribution functions and multiple surfaces," in *Graphics Interface'92 Workshop on Local Illumination*, May 1992, pp. 45–52.

[20] A. Schilling, "Towards real-time photorealistic rendering: challenges and solutions," in *HWWS'97: ACM SIGGRAPH/Eurographics workshop on Graphics hardware*. ACM, 1997, pp. 7–15.

[21] M. Olano and M. North, "Normal distribution mapping." Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina., Tech. Rep. TR97-041, 1997.

[22] F. Neyret, "Modeling, animating, and rendering complex scenes using volumetric textures," *TVCG: IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 55–70, 1998.

[23] M. Toksvig, "Mipmapping normal maps," *JGT: journal of graphics, GPU, and game tools*, vol. 10, no. 3, pp. 65–71, 2005.

[24] P. Tan, S. Lin, L. Quan, B. Guo, and H.-Y. Shum, "Multiresolution reflectance filtering," in *Rendering Techniques (Eurographics Symposium on Rendering'05 Proceedings)*, 2005, pp. 111–116.

[25] C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun, "Frequency domain normal map filtering," *ACM Transactions on Graphics (SIGGRAPH'07 Proceedings)*, vol. 26, no. 3, p. 28, 2007.

[26] P. Tan, S. Lin, L. Quan, B. Guo, and H. Shum, "Filtering and rendering of resolution-dependent reflectance models," *TVCG: IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 412–425, 2008.

[27] M. Olano and D. Baker, "Lean mapping," in *I3D'10: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2010, pp. 181–188.

[28] C. Schlick, "An inexpensive BRDF model for physically-based rendering," *Computer Graphics Forum*, vol. 13, pp. 233–246, 1994.

[29] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *Computer Graphics (SIGGRAPH'77 Proceedings)*, vol. 11, no. 2, pp. 192–198, 1977.

[30] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *Computer Graphics (SIGGRAPH'81 Proceedings)*, vol. 15, no. 3, pp. 307–316, 1981.

[31] G. J. Ward, "Measuring and modeling anisotropic reflection," *Computer Graphics (SIGGRAPH'92 Proceedings)*, vol. 26, no. 2, pp. 265–272, 1992.

[32] N. L. Max, "Horizon mapping: Shadows for bump-mapped surfaces," *The Visual Computer*, vol. 4, pp. 109–117, 1988.

[33] D. Nowrouzezahrai and J. Snyder, "Fast global illumination on dynamic height fields," *Computer Graphics Forum (Eurographics Symposium on Rendering'09 Proceedings)*, Jun. 2009.

[34] W. Heidrich, K. Daubert, J. Kautz, and H.-P. Seidel, "Illuminating micro geometry based on precomputed visibility," in *SIGGRAPH'00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 455–464.

[35] M. Ashikmin, S. Premože, and P. Shirley, "A microfacet-based BRDF generator," in *SIGGRAPH'00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 65–74.

[36] T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz, "Exponential shadow maps," in *Proceedings of Graphics Interface*. Canadian Information Processing Society, 2008, pp. 155–161.

[37] F. C. Crow, "Shadow algorithms for computer graphics," *Computer Graphics (SIGGRAPH'77 Proceedings)*, vol. 11, no. 2, pp. 242–248, 1977.

[38] L. Williams, "Casting curved shadows on curved surfaces," *Computer Graphics (SIGGRAPH'78 Proceedings)*, vol. 12, no. 3, pp. 270–274, 1978.

[39] T. Lokovic and E. Veach, "Deep shadow maps," in *SIGGRAPH'00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 385–392.

[40] T.-Y. Kim and U. Neumann, "Opacity shadow maps," in *Rendering Techniques (Eurographics Workshop on Rendering'01 Proceedings)*. Springer-Verlag, 2001, pp. 177–182.

[41] J.-M. Hasenfratz, M. Lapierre, N. Holzschuch, and F. Sillion, "A survey of real-time soft shadows algorithms," *Computer Graphics Forum (Eurographics'03 State of The Art Reports)*, vol. 22, no. 4, pp. 753–774, dec 2003.

[42] D. Scherzer, M. Wimmer, and W. Purgathofer, "A survey of real-time hard shadow mapping methods," *Computer Graphics Forum (Eurographics'10 State of the Art Reports)*, May 2010.

[43] W. Donnelly and A. Lauritzen, "Variance shadow maps," in *I3D'06: ACM SIGGRAPH Symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 161–165.

[44] T. Annen, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, "Convolution shadow maps," in *Rendering Techniques (Eurographics Symposium on Rendering'07 Proceedings)*. Eurographics, 2007, pp. 51–60.

[45] A. Lauritzen and M. McCool, "Layered variance shadow maps," in *Proceedings of Graphics Interface*. Canadian Information Processing Society, 2008, pp. 139–146.

[46] T. Annen, Z. Dong, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, "Real-time all-frequency shadows in dynamic scenes," *ACM Trans. Graph.*, vol. 27, pp. 34:1–34:8, 2008.

[47] M. Salvi, *ShaderX 6.0 – Advanced Rendering Techniques*. Charles Rivers Media, 2008, ch. 4.3, pp. 257–274.

[48] A. A. Apodaca and L. Gritz, *Advanced RenderMan: Creating CGI for Motion Picture*. Morgan Kaufmann Publishers Inc., 1999, ch. 7, pp. 159–182.

[49] ——, *Advanced RenderMan: Creating CGI for Motion Picture*. Morgan Kaufmann Publishers Inc., 1999, ch. 11, pp. 263–280.

[50] D. Peachey, "Building procedural textures," in *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., 2002, ch. 2, pp. 7–94.

[51] M. Olano, B. Kuehne, and M. Simmons, "Automatic shader level of detail," in *HWWS'03: ACM SIGGRAPH/Eurographics conference on Graphics hardware*. Eurographics Association, 2003, pp. 7–14.

[52] F. Pellacini, "User-configurable automatic shader simplification," in *ACM Transactions on Graphics (SIGGRAPH'05 Proceedings)*. ACM, 2005, pp. 445–452.

[53] K. Perlin, "An image synthesizer," *Computer Graphics (SIGGRAPH'85 Proceedings)*, vol. 19, no. 3, pp. 287–296, 1985.

[54] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. Ebert, J. Lewis, K. Perlin, and M. Zwicker, "State of the art in procedural noise functions," in *Computer Graphics Forum (Eurographics'10 State of the Art Reports)*, May 2010.

[55] A. Norton, A. P. Rockwood, and P. T. Skolmoski, "Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space," *Computer Graphics (SIGGRAPH'82 Proceedings)*, vol. 16, no. 3, pp. 1–8, 1982.

[56] J. C. Hart, N. Carr, M. Kameya, S. A. Tibbitts, and T. J. Coleman, "Antialiased parameterized solid texturing simplified for consumer-level hardware implementation," in *HWWS'99: ACM SIGGRAPH/Eurographics workshop on Graphics hardware*. ACM, 1999, pp. 45–53.

[57] S. Worley, "Advanced antialiasing," in *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., 2002, ch. 5, pp. 157–176.

[58] S. Bergner, T. Moller, D. Weiskopf, and D. J. Muraki, "A spectral analysis of function composition and its implications for sampling in direct volume visualization," *TVCG: IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1353–1360, 2006.

[59] F. K. Musgrave, "Fractal solid textures: Some examples," in *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., 2002, ch. 15, pp. 447–487.

[60] K. Perlin and E. M. Hoffert, "Hypertexture," *Computer Graphics (SIGGRAPH'99 Proceedings)*, vol. 23, no. 3, pp. 253–262, 1989.

[61] D. Hinsinger, F. Neyret, and M.-P. Cani, "Interactive animation of ocean waves," in *ACM-SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2002, July, 2002*, Jul. 2002, pp. 161–166.

[62] R. B. V. Horn III and G. Turk, "Antialiasing procedural shaders with reduction maps," *TVCG: IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 539–550, 2008.

[63] E. Bruneton, F. Neyret, and N. Holzschuch, "Real-time realistic ocean lighting using seamless transitions from geometry to BRDF," *Computer Graphics Forum (Eurographics'10 Proceedings)*, vol. 29, no. 2, 2010.

[64] H. Wu, J. Dorsey, and H. E. Rushmeier, "Characteristic point maps," *Computer Graphics Forum (Eurographics Symposium on Rendering'09 Proceedings)*, vol. 28, no. 4, pp. 1227–1236, 2009.

[65] B. G. Becker and N. L. Max, “Smooth transitions between bump rendering algorithms,” *SIGGRAPH’93*, vol. 27, pp. 183–190, 1993.

[66] V. Timonen and J. Westerholm, “Scalable height field self-shadowing,” *Computer Graphics Forum (Eurographics’10 Proceedings)*, vol. 29, no. 2, May 2010.

[67] M. Oren and S. K. Nayar, “Generalization of lambert’s reflectance model,” in *SIGGRAPH’94*. ACM Press/Addison-Wesley Publishing Co., 1994, pp. 239–246.

[68] S. H. Westin, J. R. Arvo, and K. E. Torrance, “Predicting reflectance functions from complex surfaces,” *Computer Graphics (SIGGRAPH’92 Proceedings)*, vol. 26, no. 2, pp. 255–264, 1992.

[69] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, “A practical model for subsurface light transport,” in *ACM Transactions on Graphics (SIGGRAPH’01 Proceedings)*. ACM, 2001, pp. 511–518.

[70] R. L. Cook, “Shade trees,” *Computer Graphics (SIGGRAPH’84 Proceedings)*, vol. 18, no. 3, pp. 223–231, 1984.

[71] F. Policarpo, M. M. Oliveira, and a. L. D. Comba, Jo “Real-time relief mapping on arbitrary polygonal surfaces,” *ACM Transactions on Graphics (SIGGRAPH’05 Proceedings)*, vol. 24, no. 3, pp. 935–935, 2005.

[72] N. Tatarchuk, “Dynamic parallax occlusion mapping with approximate soft shadows,” in *I3D’06: ACM SIGGRAPH Symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 63–69.

[73] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe, and J. Snyder, “Silhouette clipping,” in *SIGGRAPH’00*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 327–334.

[74] S. Jeschke, S. Mantler, and M. Wimmer, “Interactive smooth and curved shell mapping,” in *Rendering Techniques (Eurographics Symposium on Rendering’07 Proceedings)*. Eurographics, 6 2007, pp. 351–360.

[75] J. Amanatides, “Algorithms for the detection and elimination of specular aliasing,” in *Proceedings of Graphics interface*. Morgan Kaufmann Publishers Inc., 1992, pp. 86–93.

[76] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum, “View-dependent displacement mapping,” *ACM Transactions on Graphics (SIGGRAPH’03 Proceedings)*, vol. 22, no. 3, pp. 334–339, 2003.

[77] P. Cignoni, C. Montani, and R. Scopigno, “A comparison of mesh simplification algorithms,” *Computers & Graphics*, vol. 22, pp. 37–54, 1997.

[78] D. Luebke, “A survey of polygonal simplification algorithms,” Chapel Hill, NC, USA, Tech. Rep., 1997.

[79] R. Pajarola and E. Gobbetti, “Survey of semi-regular multiresolution models for interactive terrain rendering,” *Visual Computer*, vol. 23, no. 8, pp. 583–605, 2007.

[80] J. Jansen and L. Bavoil, “Fourier opacity mapping,” in *I3D’10: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 2010, pp. 165–172.