Filtering Non-Linear Transfer Functions on Surfaces

Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, Fabrice Neyret

Filtering Non-Linear Transfer Functions on Surfaces

Eric Heitz Derek Nowrouzezahrai Pierre Poulin Fabrice Neyret

Abstract—Applying non-linear transfer functions and look-up tables to procedural functions (such as noise), surface attributes, or even surface geometry are common strategies used to enhance visual detail. Their simplicity and ability to mimic a wide range of realistic appearances have led to their adoption in many rendering problems. As with any textured or geometric detail, proper filtering is needed to reduce aliasing when viewed across a range of distances, but accurate and efficient transfer function filtering remains an open problem for several reasons: transfer functions are complex and non-linear, especially when mapped through procedural noise and/or geometry-dependent functions, and the effects of perspective and masking further complicate the filtering over a pixel's footprint. We accurately solve this problem by computing and sampling from specialized filtering distributions on the fly, yielding very fast performance. We investigate the case where the transfer function to filter is a color map applied to (macroscale) surface textures (like noise), as well as color maps applied according to (microscale) geometric details. We introduce a novel representation of a (potentially modulated) color map's distribution over pixel footprints using Gaussian statistics and, in the more complex case of high-resolution color mapped microsurface details, our filtering is view- and light-dependent, and capable of correctly handling masking and occlusion effects. Our approach can be generalized to filter other physical-based rendering quantities. We propose an application to shading with irradiance environment maps over large terrains. Our framework is also compatible with the case of transfer functions used to warp surface geometry, as long as the transformations can be represented with Gaussian statistics, leading to proper view- and light-dependent filtering results. Our results match ground truth and our solution is well suited to real-time applications, requires only a few lines of shader code (provided in supplemental material), is high performance, and has a negligible memory footprint.

Index Terms—LOD, procedural texture, noise, Gaussian statistics

1 Introduction

Procedural textures are a popular approach for adding detail to 3D objects with a long-standing history [1], [2]. Modern graphics hardware allows on-the-fly evaluation of procedural texture functions with easy integration into, e.g., shader-based pipelines, requiring little additional memory usage. Another benefit of these approaches is their ability to cover a wide range of appearance variations with a small set of parameters.

As with standard textures, procedural textures require proper filtering to reduce aliasing, e.g., when viewed from varying distances or angles. Mipmapping [3] is a common texture prefiltering approach; however, it cannot apply to procedural textures where texels are evaluated on the fly rather than stored in memory. Currently, the only way to accurately filter arbitrary procedural textures is with numerical integration. The cost of this solution grows not only with the cost of evaluating the underlying procedural function but also with the number of samples which, in turn, grows with the size of the filter. As such, pixels with larger texture footprints require more integration samples, incurring a non-uniform filtering cost in image space, which is unacceptable for real-time rendering.

Lagae et al. [4] recently solve the problem of filtering procedural *noise* functions $r(x)$ by computing the filtered value $\int r(x)dx$ directly from properties of r's generating process, thus avoiding numerical integration. However, it is less common to apply noise directly as a texture. Instead, noise and other procedural processes are often mapped through a *transfer function* in order to obtain the final procedural texture [5], which can, in turn, be applied as an albedo map atop a surface or according to geometric properties of, e.g., some fine-scale microsurface. A *color map* $C(x)$ is a transfer function mapping grayscale values to colors.

Anti-aliasing color mapped procedural textures requires integrating the color map "driven" by a procedural function: $\int C(f(x))dx$. Since color maps are often non-linear, the naïve solution of color mapping the filtered/mipmapped driving function (i.e., $C(\int f(x)\,dx)$) is not valid in general. According to recent surveys [4], [6], accurate and efficient filtering of this composite function is an open problem.

We present the first method to accurately and efficiently filter color mapped textures (Section 4). Our method is several orders of magnitude faster than numerical integration, has cost independent of footprint size, and accurately filters across all scales. Our solution is exact when $f(x)$ is a noise function (e.g.,

- *E. Heitz and F. Neyret are with INRIA-LJK (Université de Grenoble and CNRS). E-mail: {Eric.Heitz, Fabrice.Neyret}@inria.fr.*
- *D. Nowrouzezahrai and P. Poulin are with LIGUM, Dept. I.R.O., Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal, Québec, Canada H3C 3J7. E-mail: {derek, poulin}@iro.umontreal.ca.*

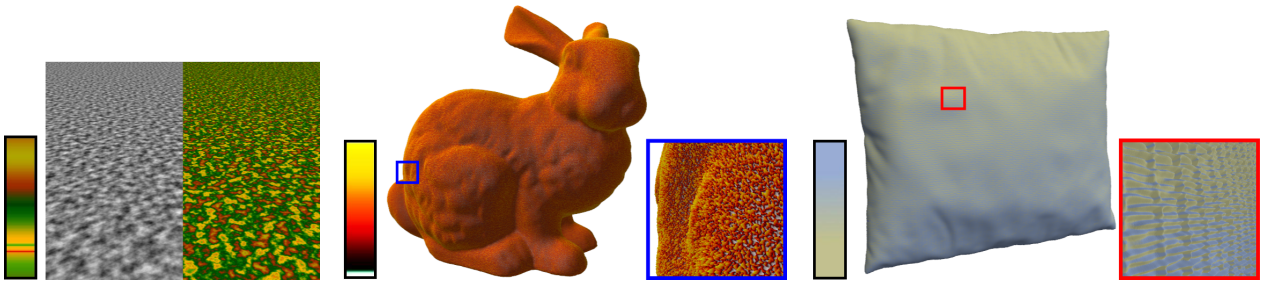


Fig. 1. An anti-aliased procedural texture produced by applying a color map to noise (left), introducing structure that cannot be generated using only the noise function. Meshes with tessellated color mapped procedural microsurface details. The color map is applied to the microsurface heights (middle) and orientations (right) and filtered appropriately according to view-dependent masking and shading effects.

$f = r$), as these processes have Gaussian statistics [4], and it also approximates filtering with non-Gaussian driving functions.

High-resolution textures traditionally augment the apparent detail of coarser underlying geometry. Given the discrepancy between texture and geometry resolutions, filtering textures while ignoring the underlying geometric variation is a suitable simplification. However, with the onset of programmable tessellation units, GPUs can now dynamically generate sub-pixel geometric detail on the fly, eliminating the texture/geometry resolution discrepancy. Given this, texture-level detail can now become correlated with the underlying geometry, and must be filtered in tandem with this geometry.

We extend our solution to two instances of joint texture-geometry filtering (Section 5): color maps correlated to heights, and to local orientations, of the underlying microdetail geometry. We note that, with joint texture-geometry filtering, the filtered result accounts for occlusion and masking (i.e., it is both view- and light-dependent). Our filtering strategy can be extended to transfer functions that map geometry to physical quantities other than albedo (Section 5.3) and the individual solutions can easily be combined linearly or non-linearly (Section 6) for greater flexibility.

Our general approach can additionally be applied to transfer functions (Section 7) that modify the microgeometry of a surface, leading to the same efficient and correct view- and light-dependent filtering of detailed shading.

2 Previous Work

For an in-depth study of procedural noise usage and filtering, we refer readers to recent surveys [4], [6]. We instead focus on solutions to filtering color mapped textures and geometry-correlated textures. We assume that the driving function f is either tabulated or a procedural noise (i.e., $f = r$).

Filtering Color Mapped Textures: Shader-simplification [7], [8], [9] progressively blends evaluated shader colors with an average shader value based on an analysis of the shader's procedural shade tree. In some cases, blending is applied once the maximal frequency of the procedural shader surpasses the pixel sampling rate. While high frequencies should ideally be filtered progressively, these methods attenuate all frequencies of the procedural shader in parallel. This results in a trade-off between anti-aliasing quality and features preservation.

Standard color map filtering uses mipmaps [3], [10] that store average color map values over an interval. An approximate interval is chosen, according to pixel footprint size and color map gradient information, to sample the mipmap. Hart et al. [11] refine this approach, approximating the integration domain with a 1^{st}-order approximation of the procedural texture, that is only valid for small footprint sizes. Lagae et al. [12] compute the spectrum of procedural noise functions, and analytically estimate the variance lost due to filtering. These methods are limited to box prefiltering.

Worley [13] uses an adaptive numerical integration scheme based on heuristic bounds of the spectrum of the procedurally driven color map. Better heuristics [14] can improve this approach, however even an adaptive numerical scheme cannot scale to the demands of interactive rendering algorithms.

We refer the reader to the independent and concurrent work of Hadwiger et al. [15] on using distribution representations for image processing. We instead focus on high-performance texture and geometry filtering using specialized Gaussian representations in the context of a real-time rendering system.

Filtering Color Mapped Surfaces: Few methods address the problem of applying procedural color maps to surfaces according to underlying properties such as normals, reflectance, or occlusion.

Wu et al. [16], [17] use characteristic point maps to find view-dependent correlations between procedural color maps and surface structures. To our knowledge their method is the most general reflectance filtering method but its memory usage and precomputation times preclude its application to truly dynamic textures.

Heitz and Neyret [18] analytically solve the special case where the color map is applied to the height of a Gaussian microsurface (Section 3). Their approach efficiently reconstructs complex view-dependent effects, however, it is restricted to only sigmoid-based color maps and height mapping. We present a more general solution for procedurally driven color maps (Section 4), procedurally driven height and local-orientation correlated color maps of *arbitrary* form (Section 5), and any combination of these approaches (Section 6). We finally generalize to height transfer functions by applying the same methodology to the transfer function's Gaussian statistics (Section 7).

3 Preliminaries and Overview

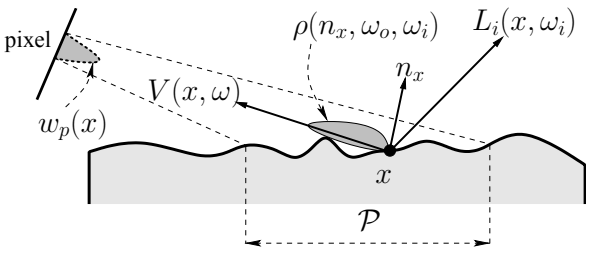


Fig. 2. The geometry of our problem. We use Gaussian statistics to filter color mapped textures and microsurface detail at all scales.

Figure 2 illustrates the geometry of our problem and Table 1 lists the key mathematical notations. We define the observed pixel intensity I, reflected by an ensemble of surface elements x towards the eye, according to the following local illumination model:

$$I = \frac{\int_{\mathcal{P}} L_i(x,\omega_i)\, C(x)\, \rho(n_x,\omega_o,\omega_i)\, V_o(x)\, V_i(x)\, w_P(x)\; dx}{\int_{\mathcal{P}} V_o(x)\, w_P(x)\; dx} \tag{1}$$

where n_x is the microsurface normal at x, $V_o(x) = V(x,\omega_o)\,\max(\cos\theta_o, 0)$, and $V_i(x) = V(x,\omega_i)\,\max(\cos\theta_i, 0)$. We only consider a single directional light source and integration over incident directions is required to handle more complex lighting.

It is clear from Equation (1) that the observed pixel intensity I depends on a complex interplay between geometry, reflectance, incident lighting, and the color map. While previous approaches have addressed the sub-problem of filtering the geometric and radiometric quantities in Equation (1), the manner in which the color map C is filtered across the pixel footprint has only recently been considered by Heitz and Neyret [18]. We briefly review their work before identifying our more complete treatment of this problem.

Symbol	Description
$\mathcal{P}$	Surface elements x that project to a fixed pixel P
$\mathcal{H}$	Surface heights h that project to a fixed pixel P
$w_P(x)$	Filter over the pixel P footprint projected on $\mathcal{P}$
$\omega_i \quad \theta_i$	Light direction and its angle formed with x's normal
$\omega_o \quad \theta_o$	View direction and its angle formed with x's normal
$V(x,\omega)$	Line-of-sight visibility of ray from x towards ω
$L_i(x,\omega_i)$	Incident radiance at x from light direction ω_i
$C(x; f(\cdot))$	Color map generated according to function f (e.g., color mapped noise, height-dependent color, etc.)
$\rho(n_x,\omega_o,\omega_i)$	Bi-directional reflectance distribution function (BRDF)

TABLE 1
The notation used throughout the paper.

Existing Work on Color Map Filtering: Heitz and Neyret [18] filter color maps applied to Gaussian microsurface heightfields $h(x)$, where the color map is a very specific function of the height, $C(x; f(\cdot)) \approx C(h(x))$, detailed below.

With a microsurface height distribution $p_h(h)$, projected onto the surface, of zero mean and variance σ_h^2, $p_h(h) = \mathcal{N}(0, \sigma_h^2)$, Heitz and Neyret apply an analytic expression for the visibility as

$$V(x,\omega) = \left[\int_{-\infty}^{h(x)} p_h(h')\, dh'\right]^{\Lambda(\omega)} = P_h(h(x))^{\Lambda(\omega)}\ , \tag{2}$$

where $P_h(h) = 1 - \frac{1}{2}\operatorname{erfc}\left(h/(\sqrt{2}\sigma_h)\right)$ is the cumulative distribution function of the microsurface heights and

$$\Lambda(\omega) = \frac{1}{\sqrt{2\pi}}\frac{\sigma_s}{\cot\theta_i}\exp\left(-\frac{\cot^2\theta_i}{2\sigma_s^2}\right) - \frac{1}{2}\operatorname{erfc}\left(\frac{\cot\theta_i}{\sqrt{2}\sigma_s}\right)\ ,$$

where $\cot\theta_i$ is the slope of the incident direction, and σ_s is the standard deviation of the microsurface slopes in direction ω.

If C is the combination of a base color c_0 and a color c_1 that is scaled according to the microsurface height (namely, $C(h) = c_0 + c_1\, P_h(h)$), then Heitz and Neyret show (using Equation (2)) that the view-occluded and light-masked filtered color is

$$\begin{aligned}\overline{C}(x,\omega_i,\omega_o) &= \frac{\int_{\mathcal{H}} C(h)\, V(h,\omega_o)\, V(h,\omega_i)\, p_h(h)\, dh}{\int_{\mathcal{H}} V(h,\omega_o)\, V(h,\omega_i)\, p_h(h)\, dh} \\ &= c_0 + c_1\frac{\Lambda(\omega_o) + \Lambda(\omega_i) + 1}{\Lambda(\omega_o) + \Lambda(\omega_i) + 2}\ ,\end{aligned} \tag{3}$$

where $\mathcal{H}$ is the projected height over $\mathcal{P}$, and the mapping $h(x)$ of locations x to heights h implicitly includes the weight $w_P(x)$.

We propose several solutions which, among other things, include Heitz and Neyret's work as a special case. We do not impose *any* constraints on the form of the color map, and we support procedural color map texture filtering as well as procedural color mapped surface modulation based on height and local-orientation variations. The latter two filtering solutions properly model the effects of occlusion towards the eye and visibility towards the light. We

consider the proper, filtered shading of irradiance environment maps applied over large terrains. We also outline how to combine the three filtering methods together.

General Problem Statement and Overview: We consider each term in the integrand of Equation (1) separately and, in particular, their correlation to the *height* and *local orientation* (which is parameterized at x, and that we interchangeably call the "slope", for simplicity of writing) of the underlying Gaussian microsurface. We leverage the fact that if two functions $a(x)$ and $b(x)$ are uncorrelated over the entire domain of integration, then the integral of their product can be simplified: $\int a\,b\,dx = \int a\,dx \int b\,dx$. Our model assumes that the Gaussian microsurface heights and slopes are uncorrelated[1] [19]. We exploit this property to factor and manage separately the terms in the integrand of Equation (1).

The incident radiance L_i is independent of the surface and can be integrated separately (a common assumption in filtering methods),

$$\overline{L_i} = \int_{\mathcal{P}} L_i(x,\omega_i)\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \quad . \tag{4}$$

The visibilities to the viewer and light, $V(x,\omega_o) = V(h(x),\omega_o)$ and $V(x,\omega_i) = V(h(x),\omega_i)$, are functions of the microsurface height (see Equation (2)), but are, according to our assumptions, uncorrelated to the microsurface slope. Similarly, the BRDF $\rho(n_x,\omega_o,\omega_i)$ and clamped cosine terms ($\max(\cos\theta_o,0)$ and $\max(\cos\theta_i,0)$) depend on the microsurface slope, but remain uncorrelated to the microsurface height.

The only remaining term of interest in the integrand of Equation (1) is the color map and, here, we decompose it into three components: a term that is completely uncorrelated to the microsurface (but still potentially driven by an abstract function f), $C(x;f(\cdot)) = C_0(f(x))$; a term that depends only on the microsurface heights, $C(h(x)) = C_h(x)$; and a term that depends only on the microsurface local orientations/slopes, $C(n_x) = C_s(x)$. Figure 3 illustrates a diagrammatic example of these three terms.

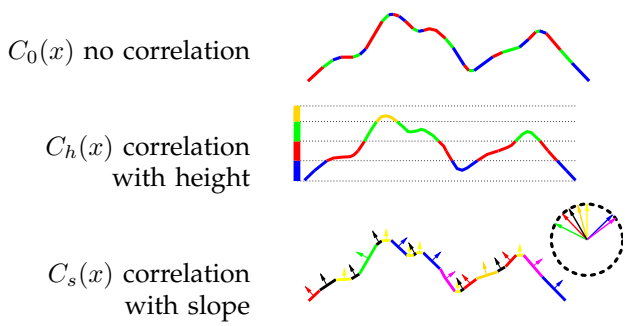


Fig. 3. We decompose the color map into three independent components.

1. Or that correlation is sufficiently small and can be neglected.

We reduce the general color map filtering problem to the problem of filtering each of these three types of color maps. The remainder of the paper is dedicated to solving Equation (1) in the context of these three cases (Sections 4 and 5), and how to combine these solutions to handle color maps composed of combinations of each three base cases (Section 6). The approach is finally generalized to geometric transfer functions (Section 7).

Note that filtering color maps of the form C_0 corresponds to the long-standing problem of filtering procedurally driven textures; we solve it in Section 4.

4 Filtering Color Mapped Textures

Even filtering the simplest instance of a color map function, $C(x;f(\cdot)) = C_0(f(x))$, that does not depend on any microsurface attributes, is an open problem in computer graphics. This scenario occurs when procedurally generated textures are used to "drive" lookups into a complex color map.

In this isolated case, Equation (1) can be simplified, exploiting the absence of correlation between the color map (and incident light) and the remaining terms in the integrand, as

$$I = \overline{L_i} \underbrace{\left[\frac{\int_{\mathcal{P}} C_0(f(x)) w_P dx}{\int_{\mathcal{P}} w_P\, dx}\right]}_{\overline{C_0}} \left[\frac{\int_{\mathcal{P}} \rho(n_x) V_o(x) V_i(x)\, w_P\, dx}{\int_{\mathcal{P}} V_o(x)\, w_P\, dx}\right], \tag{5}$$

where $\overline{C_0}$ is the average color over the pixel footprint. We omit parameters from the BRDF and footprint weight for conciseness.

Computing $\overline{C_0}$ or, in other words, the texture of f color mapped through C over the pixel footprint, requires solving the following integral at every pixel:

$$\overline{C_0} = \int_{\mathcal{P}} C_0(f(x))\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \quad , \tag{6}$$

where w_P is the filter defined over the pixel footprint projected onto $\mathcal{P}$ (for which common choices are box or Gaussian filters).

The integral in Equation (6) can be interpreted as a combination of the values contained in the color map C_0 weighted by the filter w_P, and their presence in f. As such, Equation (6) can be formulated as an inner product (expressed with angled bracket notation):

$$\overline{C_0} = \int_{-\infty}^{\infty} C_0(v)\, D_f(\mathcal{P},v)\, dv = \langle C_0, D_f(\mathcal{P},\cdot)\rangle \quad , \tag{7}$$

where $D_f(\mathcal{P},v)$ is the distribution of values in f over $\mathcal{P}$, weighted by w_P, which we call the *filtering distribution*.

To efficiently compute Equation (7) we seek a representation of the distribution $D_f(\mathcal{P},v) > 0$ that will facilitate the evaluation of the inner product. Such a representation should be *scalable*, meaning that it

can be computed with a memory footprint and a computational complexity independent of the size of the filter extent of w_P.

4.1 Determining the Filtering Distribution

The non-negative filtering distribution $D_f(\mathcal{P}, v)$ can be interpreted as a normalized histogram (i.e., $\int D_f(\mathcal{P}, x')dx' = 1$).

The filtering distribution $D_f(\mathcal{P}, v)$ has a dimensionality that grows with the number of parameters that are used to describe f, and depends on $\mathcal{P}$, w_P, and the type of these parameters. For arbitrary color maps and grayscale texture functions, the filtering distribution may have very high dimensionality. No single representation can be used to exactly describe all possible filtering distributions, let alone doing so in a manner that is both memory efficient and suitable for rapid computation of the inner product in Equation (7).

At a high level, in cases where f is a precomputed/pretabulated texture, D_f may also be precomputed/pretabulated. Similarly, if f is constructed, e.g., procedurally, then it may be possible to construct D_f from the process that generated f. We note that the choice of the filter w_P influences the form of D_f. For instance, mipmapping of an unfiltered D_f would correspond to having a box filter as w_P.

We will briefly discuss conditions under which the filtering distribution may be exactly representable (in a reasonable amount of time and memory), or approximated, before investigating a specific solution that exploits Gaussian statistics (Section 4.2).

Exact Solutions: As mentioned above, for tabulated f (and, specifically, C_0 and f that are low dimensional), D_f may be represented with only a few parameters: e.g., when f is a grayscale function with a small number of entries $v \in \{v_1, \ldots, v_n\}$. Here, the D_f histogram can be discretized and precomputed. If C_0 is also coarsely discretized, $C_0 = \sum_i \lambda_i \delta_{v_i}$, D_f needs only be evaluated at the v_i samples.

This idea generalizes to the case where both C_0 and D_f are represented with a finite weighted combination of basis functions $\{a\}$ and $\{b\}$: $C_0(v) = \sum_i \alpha_i a_i(v)$ and $D_f(\mathcal{P}, v) = \sum_j \beta_j b_j(v)$. In this general case the inner product in Equation (7) reduces to

$$\langle C_0, D_f(\mathcal{P}, \cdot)\rangle = \sum_i \sum_j \alpha_i \, \beta_j \, \langle a_i, b_j\rangle \ , \qquad (8)$$

where the inner product of basis function pairs $\langle a_i, b_j\rangle$ can be (pre)computed analytically. Exact analytic solutions may be possible depending on the choice of the basis functions; for example, precomputed radiance transfer [20], [21] considers the special case where the basis function sets used to represent both functions are identical (i.e., $\{a\} = \{b\}$) and orthonormal (i.e., $\langle a_i, b_j\rangle = \delta_{ij}$, where δ_{ij} is the Kroenecker delta function).

Approximate Solutions: When f cannot be exactly expressed in a finite basis set, but instead can be approximated, e.g., with a Taylor expansion (see [6] for examples of other approximations), we may be able to leverage properties of the representation in order to efficiently approximate D_f. For example, when f is a procedural function, the processes used to generate f may have statistical properties that can be used to define the filtering distribution: e.g., noise functions can be defined as processes that produce Gaussian distributions [4]. The statistical distribution of the process can be used as an approximate substitute for the distribution D_f of the considered instance of the process. Note that, in this case, the distribution differs slightly from the correct distribution since the statistics of an instance never perfectly match the statistics of the generative process. However, when the number of considered samples increases (and as the size of the filter w_P increases), the statistics of the instance are well approximated by the statistics of the process and converge toward it.

We will exploit these observations next in Section 4.2 in order to devise our specific filtering solution.

4.2 Color Mapped Gaussian Distributions

We propose a solution for cases where $f(x)$ is a procedural (e.g., noise) function generated through a Gaussian process. A broad set of procedural functions commonly used in computer graphics fall into this category (see [4]). We are further motivated by the compactness of Gaussian representations—only two parameters, mean and variance, are necessary to fully describe them—and the fact that Gaussians accurately approximate an important class of real-world distributions. As such, these advantages make Gaussian distributions a good choice for a generic lightweight representation.

What's more, the inner product of a function C_0 and a Gaussian with mean $\bar{f}$ is the convolution $C_0 * \mathcal{N}(0, \sigma_f^2)$ evaluated at the mean,

$$\langle C_0, \mathcal{N}(\bar{f}, \sigma_f^2)\rangle = \left[C_0 * \mathcal{N}(0, \sigma_f^2)\right](\bar{f}) \ . \qquad (9)$$

In practice, this allows us to precompute the convolutions of C_0 and Gaussian kernels $\mathcal{N}(\bar{f}, \sigma_f^2)$ with standard deviation values in the interval $[0, \sigma_{\max}]$, where $\sigma_{\max}$ is a user-parameter we set according to the noise process. We store these preconvolutions in a 2D texture indexed by $\bar{f}$ and σ_f. At run-time, a shader efficiently computes $\bar{f}$ and σ_f *over the footprint* $\mathcal{P}$ of the procedural (e.g., noise) function, and uses these parameters to sample $\bar{C}_0$ (see Figure 4).

Thus, the color map filtering problem is reduced to the efficient computation of the mean and standard deviation of f within the footprint $\mathcal{P}$. For precomputed/tabulated f, we need only precompute the first two moments of f, $\bar{f}$ and $\overline{f^2}$, at each level of detail

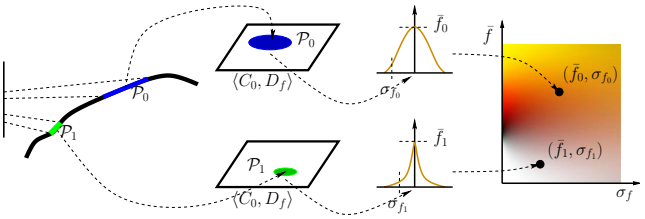
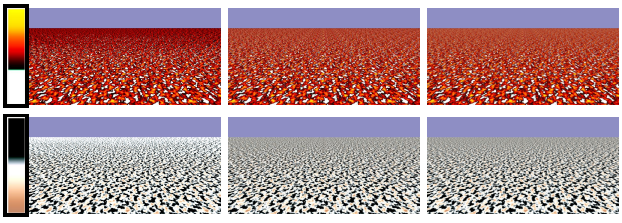


Fig. 4. We prefilter the color map applied to a procedural function (e.g., noise). We sample specially constructed filtering distributions (right) using the function's statistics over the surface patch.



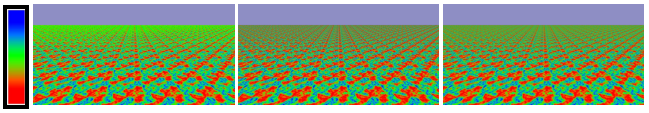
Naïve filtering Ground truth Our result

Fig. 5. Filtering color mapped textures that cannot be created using only the application of a procedural (noise) function.

in a mipmap hierarchy of the original tabulated f. We compute the variance of f after linear texture interpolation of the first two moments, sampled from the appropriate level of detail in the mipmap hierarchy: $\sigma_f^2 = \bar{f}^2 - \bar{f}$. Note that this method is also compatible with anisotropic texture filtering methods (i.e., more complex w_P). For procedural (e.g., noise) functions f, such as Perlin noise [1], we precompute the noise standard deviation for different levels of detail and store it in a 1D texture. More sophisticated noise functions such as Gabor noise [12] allow for an analytical evaluation of the noise variance in the spectral domain. Note that the idea of preconvolving a signal with a Gaussian is not new. Previous work based on this technique are discussed in [6].

We have detailed a method for computing $\overline{C_0}$, and thus Equation (5), under the Gaussian statistics assumption. This method allows us, for the first time, to accurately filter a color mapped procedural function at very high framerates. Figure 5 compares our color map filtering method to a super-sampled ground-truth simulation, as well as a standard naïve method of sampling the color map with the filtered driving function f (i.e., $C_0(\int f(x)dx)$).

Our filtering results closely match ground truth and have performance roughly equivalent to the naïve mipmapping solution. Figure 6 illustrates an example of explicit data with non-Gaussian statistics. Here, we approximate the histogram of the data (right; blue) with a Gaussian (right; red) and the resulting simplification still yields an accurate result. Some results also appear in Section 1 of the supplemental document associated with this paper.



Naïve filtering Ground truth Our result

Fig. 6. The color map (left) is applied to our filtering method with a Gaussian approximation (right), which still closely matches ground truth computed with the original statistics (middle).

5 Filtering Color Mapped Surfaces

If the color is correlated to a surface attribute, then Equation (5) is no longer a valid solution to Equation (1) and we require new specialized solutions that depend on the type of correlation.

We will isolate and discuss solutions to two cases: first, when $C(x)$ is correlated to the microsurface heights (Section 5.1) and, second, when it is correlated to the microsurface slopes (Section 5.2). The latter can be applied to shading with irradiance environment maps (Section 5.3).

Finally, we will discuss how these individual solutions, as well as the texture filtering solution presented in Section 4, can be combined to handle more general color maps (Section 6).

5.1 Height-correlated Color Filtering

When the color is correlated to the microsurface heights, namely $C(x; f(\cdot)) \approx C(h(x)) = C_h(x)$, then Equation (1) can be factored according to terms that depend on the height and those that do not:

$$I = \overline{L_i} \underbrace{\left[\frac{\int_{\mathcal{P}} C_h(x)\, V(x,\omega_o)\, V(x,\omega_i)\, w_P\, dx}{\int_{\mathcal{P}} V(x,\omega_o)\, w_P\, dx} \right]}_{\overline{C_h}} \times \underbrace{\left[\frac{\int_{\mathcal{P}} \rho(n_x)\ \max(\cos\theta_i, 0)\ \max(\cos\theta_o, 0)\, w_P\, dx}{\int_{\mathcal{P}} \max(\cos\theta_o, 0)\, w_P\, dx} \right]}_{\overline{\rho}}, \tag{10}$$

where $\overline{\rho}$ is the filtered reflectance. We simplify the problem of solving Equation (10)—and thus, Equation (1) in this special case—to that of computing $\overline{C_h}$. Solving the reflectance filtering problem is outside the scope of our work; we employ one of the simpler techniques described in the recent survey [6].

Motivated by the flexibility of Gaussian statistics, which we already leveraged in Section 4.2, we will assume that the height density $p_h(h)$ of our microsurface geometry is formed according to a zero-mean Gaussian process: $\mathcal{N}(0, \sigma_h^2)$. In Section 9 we show that GPU tessellation shaders can be used to procedurally

add microsurface geometry according to these same statistics.

We will now proceed to our solution to Equation (10) that extends the ideas and techniques presented earlier in Section 4 for filtering uncorrelated color mapped functions.

Filtering Color Mapped Gaussian Height Distributions: We first define the averaged shadowing over the surface footprint,

$$\overline{V}(\omega_o,\omega_i)=\int_{\mathcal{P}} V(x,\omega_o)\,V(x,\omega_i)\,w_P dx \Big/ \int_{\mathcal{P}} V(x,\omega_o)\,w_P dx\ , \tag{11}$$

which we solve for analytically (in the case of Gaussian microsurface height distributions) by substituting each visibility term in the integrands above with Equation (2)[2]:

$$\begin{aligned}\overline{V} &= \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\,P_h(h)^{\Lambda(\omega_i)}\,p_h(h)\,dh \Big/ \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\,dh \\ &= \int_{\mathcal{H}} P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h)\,dh \Big/ \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\,dh \\ &= \frac{1+\Lambda(\omega_o)}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}\ .\end{aligned} \tag{12}$$

Given the average visibility above, we can simplify $\overline{C_h}$ as follows:

$$\begin{aligned}\overline{C_h}(\omega_o,\omega_i) &= \frac{\int_{\mathcal{P}} C_h(x)\,V(x,\omega_o)\,V(x,\omega_i)\,w_P\,dx}{\int_{\mathcal{P}} V(x,\omega_o)\,w_P\,dx} \\ &= \overline{V}(\omega_o,\omega_i)\,\frac{\int_{\mathcal{H}} C_h(h)\,V(h,\omega_o)\,V(h,\omega_i)\,p_h(h)\,dh}{\int_{\mathcal{H}} V(h,\omega_o)\,V(h,\omega_i)\,p_h(h)\,dh} \\ &= \overline{V}(\omega_o,\omega_i)\int_{\mathcal{H}} C_h(h)\,D_h(\mathcal{P},h,\omega_o,\omega_i)\,dh \\ &= \overline{V}(\omega_o,\omega_i)\,\langle C_h, D_h(\mathcal{P},\cdot,\omega_o,\omega_i)\rangle\ ,\end{aligned} \tag{13}$$

where $D_h(\mathcal{P},\cdot,\omega_i,\omega_o)$ is a normalized distribution of heights over $\mathcal{P}$, and generalizes the filtering distribution idea introduced in Section 4.1 for solving the uncorrelated color map filtering problem in Equation (7). It is important to note that this new filtering distribution depends on light and view directions.

All that remains to solve Equation (10) is an accurate and computationally efficient representation of D_h, which we present below.

An Efficient Height Filtering Distribution Representation: Given Equation (13) we see that

$$D_h(\mathcal{P},h,\omega_o,\omega_i)=\frac{V(h,\omega_o)\,V(h,\omega_i)\,p_h(h)}{\int_{\mathcal{H}} V(h',\omega_o)\,V(h',\omega_i)\,p_h(h')\,dh'}\ . \tag{14}$$

After substituting Equation (2) (similarly to the development of Equation (12) from Equation (11)), we have:

$$D_h(\mathcal{P},h,\omega_o,\omega_i)=\frac{P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h)}{\int_{\mathcal{H}} P_h(h')^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h')\,dh'}\ . \tag{15}$$

Note that when the microsurface is lit and observed from directly above (i.e., head-on incidence, where $\theta_i = \theta_o = 0°$), then $\Lambda(\omega_o)+\Lambda(\omega_i) = 0$ and the height filtering distribution reduces to the microsurface height distribution: $D_h = p_h$. This is clear as occlusion (of either the light and/or the view) of the heightfield microsurface only occurs at off-normal incidence, and thus the height filtering distribution is only modulated in these circumstances.

As such, when $\Lambda(\omega_o)+\Lambda(\omega_i) > 0$, we observe empirically that D_h can be approximated very closely with a single Gaussian (see Figure 7): $D_h \approx \mathcal{N}(\mu_d,\sigma_d)$, and we fit the following non-linear functions to these parameters, starting with the mean,

$$\begin{aligned}\mu_d &= \frac{\int_{\mathcal{H}} h\,P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h)\,dh}{\int_{\mathcal{H}} P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_)]}\,p_h(h)\,dh} \\ &\approx \alpha_\mu\,\sigma_h\,\log\left(\beta_\mu\,[\Lambda(\omega_o)+\Lambda(\omega_i)]+1.0\right)\ ,\end{aligned} \tag{16}$$

and variance

$$\begin{aligned}\sigma_d^2 &= \frac{\int_{\mathcal{H}} h^2\,P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h)\,dh}{\int_{\mathcal{H}} P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\,p_h(h)\,dh} \\ &\approx \left[\frac{\sigma_h}{1.0+\alpha_\sigma\,\log\left(\beta_\sigma\,\left(\Lambda(\omega_o)+\Lambda(\omega_i)\right)+1.0\right)}\right]^2\ .\end{aligned} \tag{17}$$

We obtain the form of Equation (16) by empirically observing that: μ_d grows proportional to σ_h (leading to the multiplicative σ_h term); μ_d increases monotonically with $[\Lambda(\omega_o)+\Lambda(\omega_i)]$ with a derivative that decreases as $[\Lambda(\omega_o)+\Lambda(\omega_i)]$ increases (leading to the $\log(\text{constant}\times[\Lambda(\omega_o)+\Lambda(\omega_i)])$ term); and that $\mu_d = 0$ at $[\Lambda(\omega_o)+\Lambda(\omega_i)] = 0$ (leading to the 1.0 offset in the $\log$). We follow a similar methodology to derive the form of Equation (17).

We fit the parameters of the approximations using non-linear optimization, obtaining: $\alpha_\mu = 0.39, \beta_\mu = 4.75, \alpha_\sigma = 0.26$, and $\beta_\sigma = 1.13$. Figure 7 compares our approximations to the true values of μ_d and σ_d at several view/light directions.

At head-on incidence ($\theta_i = \theta_o = 0°$) we effectively match the expected effective height distribution, and our approximation is accurate even at grazing angles. Figure 7 also illustrates the shifting and sharpening behavior of the distribution as the view (or light) angle increases. Properly capturing this warping and shifting of the effective height distribution when the view and/or light configurations change is essential in order to properly filter the height-mapped color map. Our representation is easy and efficient to compute, requires little additional memory, and accurately captures the behavior of the ground truth effective

2. Here, we additionally change the domain of integration from the pixel footprint to the microsurface heights, absorbing the footprint weighting w_p into the microsurface height distribution $p_h(h)$.

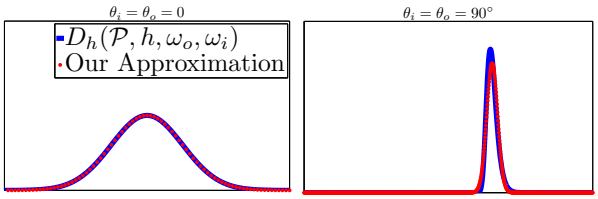


Fig. 7. Our analytic approximation of $D_h(\mathcal{P}, h, \omega_o, \omega_i)$. When $\theta_* = 0°$ (left) the distribution and our approximation match microsurface height distribution. At grazing angles, $\theta_* = 90°$ (right) the distribution is still well approximated by a Gaussian.

height distribution. Figure 8 as well as Section 2 of the supplemental document illustrate some results.

Our representation for the effective height distribution can also easily be extended to a microsurface with noise-perturbed heights. For example, instead of mapping the color map directly to the microsurface heights, we can offset the heights according to a noise function $r(x)$ and then sample the color map: $C(h(x) + r(x))$. In this case we need only modify the mean and variance of our effective height filtering distribution as $\bar{\mu}_d = \mu_d + \mu_r$ and $\bar{\sigma}_d^2 = \sigma_d^2 + \sigma_r^2$, where μ_r and σ_r are the mean and standard deviation of the noise $r(x)$. We generate a Gaussian prefiltered color map hierarchy and Equations (16) and (17) are used to sample the appropriately filtered color map value in the hierarchy at run-time[3](see Section 9).

5.2 Slope-correlated Color Filtering

When the color function depends on the slope of the surface (e.g., $C(x; f(\cdot)) = C(n_x) = C_s(x)$), then we require another factorization of Equation (1) that segments the integrand into terms that depend on the local orientation n_x and terms that do not:

$$I = \overline{L_i} \underbrace{\left[\frac{\int_{\mathcal{P}} V(x,\omega_o)\, V(x,\omega_i)\, w_P\, dx}{\int_{\mathcal{P}} V(x,\omega_o)\, w_P\, dx}\right]}_{\overline{V}} \times \left[\frac{\int_{\mathcal{P}} C_s(x)\, \rho(n_x)\, \max(\cos\theta_o, 0)\, \max(\cos\theta_i, 0)\, w_P\, dx}{\int_{\mathcal{P}} \max(\cos\theta_o, 0)\, w_P\, dx}\right].$$

This formulation is particularly difficult to solve due to potential correlations between the color map and the BRDF, and so we make an additional assumption that these two terms are uncorrelated, leading to a simplified formulation:

$$I = \overline{L_i} \times \overline{V} \times \overline{\rho} \times \underbrace{\left[\frac{\int_{\mathcal{P}} C_s(x)\, \max(\cos\theta_o, 0)\, w_P\, dx}{\int_{\mathcal{P}} \max(\cos\theta_o, 0)\, w_P\, dx}\right]}_{\overline{C_s}},$$

3. We sample with the *standard deviation* σ_d, not the variance σ_d^2.

where we re-formulate the average foreshortened color $\overline{C_s}$ as

$$\overline{C_s}(\omega_o) = \int_{\Omega_x} C(n_x)\, D_s(\mathcal{P}, n_x, \omega_o)\, dn_x = \langle C_s, D_s(\mathcal{P}, \cdot, \omega_o)\rangle, \tag{18}$$

where $D_s(\mathcal{P}, \cdot, \omega_o)$ is a normalized distribution of slopes over $\mathcal{P}$. We are following a similar methodology as in the earlier cases, and all that is required is a robust representation for D_s. Equation (18) is an accurate and computationally efficient representation of D_s.

An Efficient Slope Filtering Distribution Representation: The slope distribution of a Gaussian microsurface is itself a Gaussian with average slope $\bar{s} = (\bar{s}_x, \bar{s}_y)$ and covariance Σ. Here, $\bar{s}$ and Σ are defined in the local coordinate frame of x.

We observe empirically that the slope filtering distribution D_s shifts and stretches as ω_o varies, and it can be well approximated with a single Gaussian whose parameters can be computed directly from $\bar{s}$ and Σ. Our approximation of D_s is illustrated in Figure 9 and we include its full derivation in our supplemental document.

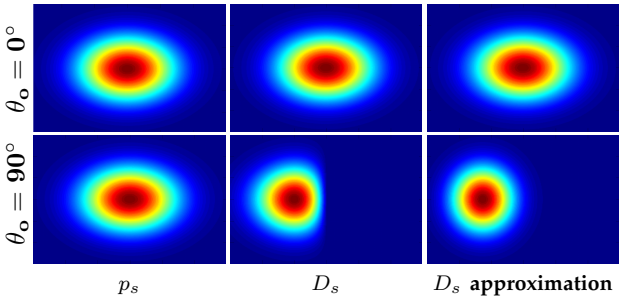


Fig. 9. (Left to right) The distribution of slopes in $\mathcal{P}$, the ground-truth view-dependent slope filtering distribution D_s, and our analytic approximation of D_s. Note that the view-dependence of D_s results in shifting and stretching of the distribution at grazing angles. Here we color code the directional probability density in angular (θ,ϕ) coordinates.

5.3 Filtering for Irradiance Environment Maps

An irradiance environment map [22] tabulates outgoing radiance for (unshadowed) Lambertian reflection from environment lighting as a function of normal orientation. Filtering diffuse shading by evaluating the diffuse BRDF with filtered normals is a common solution, but it does not correctly capture the dependence on surface microstructure that causes view-dependent effects, even in the diffuse case. We can in fact interpret an irradiance map as an orientation-correlated energy density transfer function and apply our approach to the problem of filtering diffuse reflectance from environment lighting.

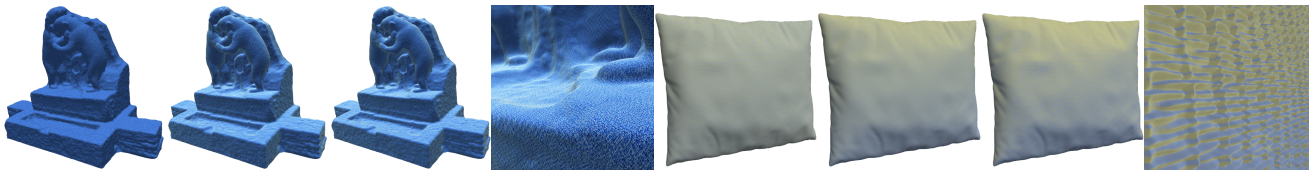


Fig. 8. View-independent filtering versus surface filtering of color mapped microsurface heights and slopes.

We apply our color mapping filtering framework to Lambertian terrains lit by an environmental lighting by treating the irradiance captured at each normal as a function applied to the surface slope, i.e., irradiance is treated as a slope color map and filtered with our framework. We note that this assumption only holds if the macrosurface is planar, as we do not consider macroscale masking/shadowing.

In this case, the local tangent frame is spatially invariant and each microslope of the surface is associated with a unique normal in world space, and thus a unique irradiance value. An example use scenario would be level-of-detail (LOD) rendering of massive terrains with irradiance mapping. Figure 10 compares our environment map filtering to traditional normal filtering and ground truth (computed with supersampling), clearly illustrating the accuracy of our approach in this context.

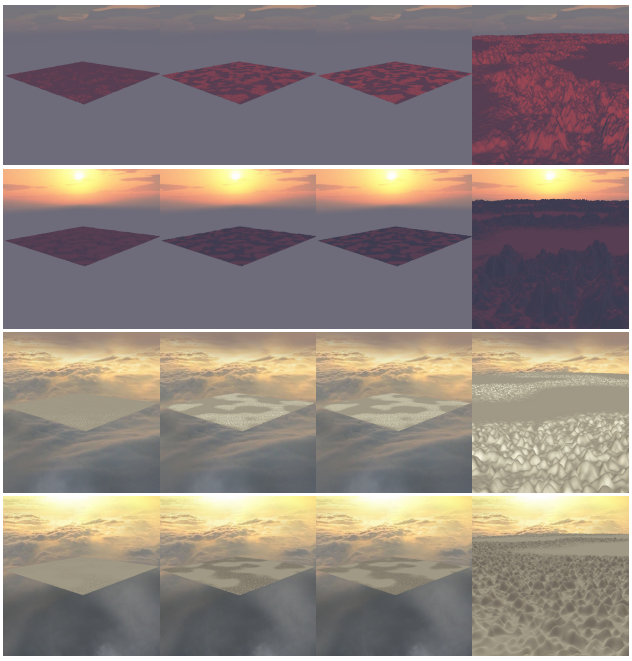


Fig. 10. Filtering diffuse reflection from irradiance environment maps on detailed microstructures. The outgoing radiance is a function of the microsurface orientations (C_s). When the predominant directional light source in the environment is behind the camera (*top of each pair of rows*), front-facing microgeometry reflects more light; when this light source is behind front-facing microgeometry (*bottom of each pair of rows*), its shading diminishes appropriately. When viewed at a distance, our filtering properly maintains this important relative intensity change as well as the spatial variation of the shading. Evaluating the irradiance map on the averaged normal does not preserve this view-dependence.

6 Combining Techniques into Solutions

Sections 4 and 5 describe methods for filtering color maps correlated to *individual* properties of the underlying microsurface geometry. In this section we investigate how to combine these individual solutions to more general color map filtering problems.

Linear Combinations: If the final color is expressed as a *linear combination* of height-correlated (Section 5.1), slope-correlated (Section 5.2), and color mapped textures (Section 4.2),

$$C = \kappa_0\ C_0 + \kappa_1\ C_h + \kappa_2\ C_s\ ,$$

then we can combine our individual solutions to solve Equation (1):

$$I = \overline{L_i}\ \overline{\rho}\ \left[\kappa_0\ \overline{C_0}\ \overline{V} + \kappa_1\ \overline{C_h} + \kappa_2\ \overline{C_s}\ \overline{V}\right]\ .$$

Non-Linear Combinations: If instead, the final output color is expressed as a *non-linear combination* of our different correlated color mappings, then the final color will be a non-linear product of the filtering results with the appropriately filtering uncorrelated terms (if any) included in the product. For example, if $C = C_0\ C_h\ C_s$ then $I = \overline{L_i}\ \overline{C_0}\ \overline{C_h}\ \overline{C_s}$.

7 Filtering Height Transfer Functions

We have demonstrated how to filter colors and textures applied on surface height maps, assuming surface distributions of heights and slopes are driven by known Gaussian statistics. We used these statistics to correctly filter view- and light-dependent surface colors at different scales. In the special case of surfaces driven by a noise function $h = f$, we showed

that the Gaussian height and slope statistics can be analytically deduced from the noise process. While interesting procedural height maps can be produced using noise functions, it remains particularly challenging to produce visually rich content exclusively using these functions.

Similar to the application of non-linear color maps to gray-scale functions, applying non-linear mapping to heights of a surface is a simple and expressive way to extend the class of representable procedurally generated height maps. Here we investigate the case where surface height maps h are mapped with a 1D spatial transfer function g applied to an existing gray-scale height function (e.g., noise) f. The surface height is then defined as $h(x) = g(f(x))$ as shown in Figure 11.

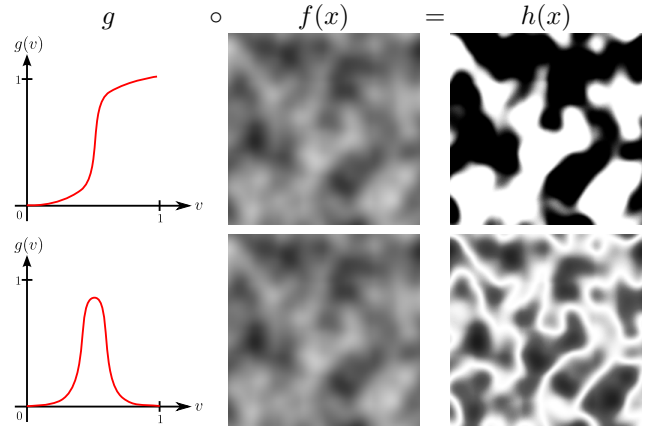


Fig. 11. The height h results from a noise pattern f mapped with a non-linear height transfer function g. These transfer functions are used to create the moss and sponge patterns shown in Figure 12.

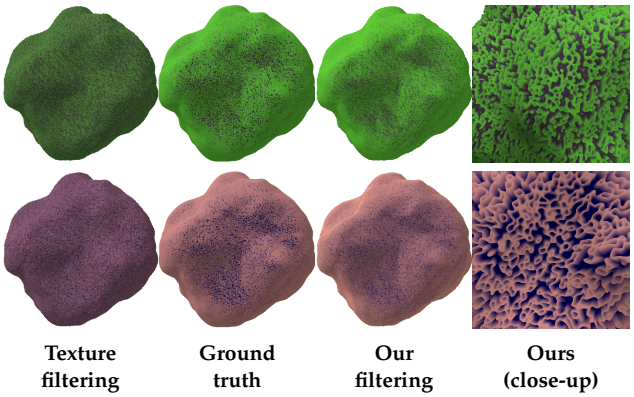


Fig. 12. View-independent texture filtering versus our surface filtering for color mapped microsurface heights, with heights transformed by the transfer functions of Figure 11 .

However, if a non-linear height transfer function g is applied to f, then the height map statistics $h = g \circ f$ are not identical to those of the underlying noise process f. The application of the transfer function g modifies the surface properties and statistics.

In order to support this different kind of height surface within our color mapping filtering framework, we need to accurately approximate its height and slope statistics, p_h and p_s, with Gaussian distributions. These can then be applied in a similar fashion as described in Section 5. As previously discussed in Sections 4.2 and 5.1, we suppose that the surface gray-scale (height) distribution D_f of f is known and follows a 1D-Gaussian distribution. Furthermore, as in Section 5.2, we suppose that the first and second moments of the gray-scale slope $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$ distribution are available.

In order to approximate the surface height distribution p_h with a Gaussian, parameterized as usual by its mean $\overline{h}$ and variance σ_h^2, we need to compute first and second moments

$$\begin{aligned}\overline{h} &= \int_{\mathcal{P}} h(x)\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \\ &= \int_{\mathcal{P}} g(f(x))\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \qquad (19)\\ \overline{h^2} &= \int_{\mathcal{P}} h^2(x)\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \\ &= \int_{\mathcal{P}} g^2(f(x))\, w_P(x)\, dx \Big/ \int_{\mathcal{P}} w_P(x)\, dx \ . \qquad (20)\end{aligned}$$

These equations resemble Equation (6) as computing the average value of a height transfer function applied to a gray-scale function is similar to computing the average color of a color map applied to a gray-scale function. Thus, we proceed in a similar fashion as presented in the color mapping case: we store first and second moments of g in a preconvolved texture parameterized by the f's statistics and we compute

$$\overline{h} = \overline{g} = \int_{-\infty}^{\infty} g(v)\, D_f(\mathcal{P}, v)\, dv = \langle g, D_f(\mathcal{P}, \cdot)\rangle \qquad (21)$$

$$\overline{h^2} = \overline{g^2} = \int_{-\infty}^{\infty} g^2(v)\, D_f(\mathcal{P}, v)\, dv = \langle g^2, D_f(\mathcal{P}, \cdot)\rangle \ . \qquad (22)$$

We use the average height value $\overline{h}$ within the filter region w_P to displace the surface during tessellation and we compute the variance as $\sigma_h^2 = \overline{h^2} - \overline{h}^2$. We approximate the PDF p_h of h (within filter region w_P) with the 1D Gaussian $\mathcal{N}(\overline{h}, \sigma_h^2)$, and we use this distribution in Equation (15).

The slopes of the surface are the gradient of the heights $\left(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}\right)$ which, when driven by the transfer function, can be expressed as, e.g.,

$$\frac{\partial h}{\partial x} = \frac{\partial g(f)}{\partial x} = g'(f)\, \frac{\partial f}{\partial x} \ . \qquad (23)$$

Section 3 assumed that the surface heights and slopes (of a Gaussian process) are uncorrelated, which leads to f and $\frac{\partial f}{\partial x}$ being uncorrelated; as such, $g'(f)$ and $\frac{\partial f}{\partial x}$ are also uncorrelated. Now we can separably integrate

the moments of $\frac{\partial h}{\partial x} = g'(f)\,\frac{\partial f}{\partial x}$; they are simply the moments of $\frac{\partial f}{\partial x}$ multiplied by the moments of g':

$$\overline{\frac{\partial h}{\partial x}} = \overline{g'}\,\overline{\frac{\partial f}{\partial x}} \tag{24}$$

$$\overline{\left(\frac{\partial h}{\partial x}\right)^2} = \overline{(g')^2}\,\overline{\left(\frac{\partial f}{\partial x}\right)^2} \tag{25}$$

$$\overline{\left(\frac{\partial h}{\partial x}\frac{\partial h}{\partial y}\right)} = \overline{(g')^2}\,\overline{\left(\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\right)}\ , \tag{26}$$

where h's average slope vector is $\left(\overline{\frac{\partial h}{\partial x}}, \overline{\frac{\partial h}{\partial y}}\right)$, and its variance and covariance are given by $\sigma_x^2 = \overline{\left(\frac{\partial h}{\partial x}\right)^2} - \overline{\frac{\partial h}{\partial x}}^2$, $\sigma_y^2 = \overline{\left(\frac{\partial h}{\partial y}\right)^2} - \overline{\frac{\partial h}{\partial y}}^2$, and $c_{xy} = \overline{\left(\frac{\partial h}{\partial x}\frac{\partial h}{\partial y}\right)} - \overline{\frac{\partial h}{\partial x}}\,\overline{\frac{\partial h}{\partial y}}$. The 2D Gaussian distribution given by these parameters corresponds to the surface slope distribution p_s and is used in the computation of the visible slope distribution D_s of Equation (18) (see our supplemental material). The moments $\overline{g'}$ and $\overline{(g')^2}$ of the derivatives of g and g', respectively, can also be similarly computed. We store the first and second moments of g' in a preconvolved texture parameterized by the statistics of f:

$$\begin{aligned}\overline{g'} &= \int_{\mathcal{P}} g'(f(x))\,w_P(x)\,dx \Big/ \int_{\mathcal{P}} w_P(x)\,dx \\ &= \int_{-\infty}^{\infty} g'(v)\,D_f(\mathcal{P},v)\,dv = \langle g', D_f(\mathcal{P},\cdot)\rangle \end{aligned} \tag{27}$$

$$\begin{aligned}\overline{(g')^2} &= \int_{\mathcal{P}} (g')^2(f(x))\,w_P(x)\,dx \Big/ \int_{\mathcal{P}} w_P(x)\,dx \\ &= \int_{-\infty}^{\infty} g'^2(v)\,D_f(\mathcal{P},v)\,dv = \langle (g')^2, D_f(\mathcal{P},\cdot)\rangle\ . \end{aligned} \tag{28}$$

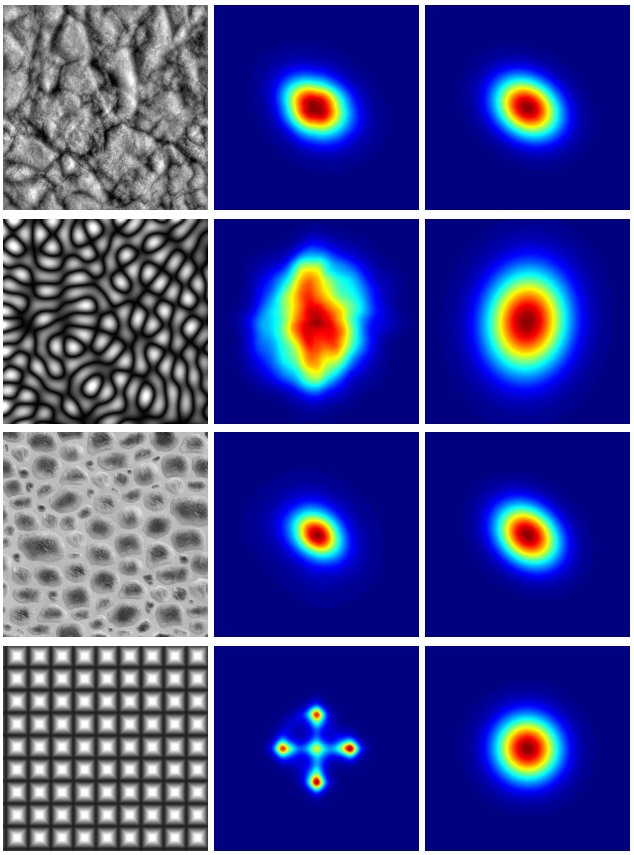
In practice we store in a preconvolved texture the moments $\overline{g}$, $\overline{g^2}$, $\overline{g'}$, and $\overline{(g')^2}$ of the height transfer function g and its derivative g'. This texture is sampled according to D_f's average and standard deviation, as with the color mapping case in Section 4. Tessellated surfaces are displaced according to $\overline{g}$ and, given $\overline{g}$ and $\overline{g^2}$, we compute the 1D Gaussian height distribution p_h. From $\overline{g'}$, $\overline{(g')^2}$, and the moments of $\frac{\partial f}{\partial x}$, we compute the 2D Gaussian slope distribution p_s. Finally, given these surface height and slope distributions, we are able to directly apply our color mapping approach as described in earlier.

Figure 12 illustrates how a finely tessellated displaced microstructure is properly shaded with our method, especially along the silhouettes and when viewed from overhead. Simply filtering the texture without the associated masking/shadowing effects generates a very different appearance than ours and the ground truth results.

8 About the Gaussian Approximation

The filtering framework described in this paper relies on several Gaussian approximations. This section discusses their validity and robustness.

Figure 13 illustrates different types of surfaces and associated Gaussian approximations. One implication of the Central Limit Theorem is that most unstructured, stochastic, fractal, or multilayered surfaces tend towards having Gaussian statistics, and therefore in many cases our Gaussian approximation is very accurate. However, the approximation breaks down in cases of repetitive, multimodal, or structured patterns, as illustrated in Figure 13 (bottom).



Microsurface **Slope distribution D_s** **Approximation G**

Fig. 13. Microsurfaces, their associated slope distributions, and our Gaussian approximations.

Error Analysis: Figure 13 illustrates the approximation inherent in our Gaussian representation, and we will detail the sensitivity of our output to the error introduced by this approximation. We have shown that the general problem of filtering color maps applied on textures or surfaces can be efficiently solved by computing the inner product of the color map C and the view-dependent distribution D of the attribute a on which the color map is applied, expressed as

$$\bar{C} = \int C(a)\,D(a)\,da, \tag{29}$$

where a can represent many different potential quantities, such as a grayscale noise value, a height, or a slope. It is important to note that this formulation, used throughout our paper, introduces no approxima-

tion (i.e., it is exact); however, to allow for fast and efficient evaluation in Sections 4, 5 and 7, we approximate D with a Gaussian, $G \approx D$ and precompute and store the inner products $\int C(a)\, G(a)\, da$ in a preconvoluted color map. This inner product can be expressed in the frequency domain, since Fourier transformation is a linear operator and thus preserves inner products:

$$\bar{C} = \int \mathcal{F}\{C\}(w)\, \mathcal{F}\{G\}(w)\, dw, \qquad (30)$$

where $\mathcal{F}$ is the Fourier transform operator. Replacing D by G preserves the value $\mathcal{F}\{G\}(0)$ of the spectrum at DC. Indeed, the DC component is the average value of G, and thus of D. In general, low frequencies with periods much larger than the standard deviation of the distribution tend to be preserved when replacing D by G. An important spectral property of the Gaussian approximation, $G \approx D$, is that *error remains primarily in higher, rather than lower, frequencies of* G. This implies that, given the inner product of C and G, the higher the frequency content of C the larger the sensitivity of the error to the Gaussian approximation. Figure 14 illustrates this behavior in the errors produced by applying color maps on slopes on surfaces with either Gaussian or non-Gaussian statistics: the result can be accurate even for highly non-Gaussian surfaces, as long as the color map has little high frequency content. Our technique is particularly well-suited to the use of gradient-based color maps, since these have zero frequency content outside of DC. If the color map is linear or affine, i.e. $C(a) = c_1 a + c_2$, then

$$\bar{C} = \int C(a)\, D(a)\, da = c_1 \int a\, D(a)\, da + c_2 \qquad (31)$$

where $\int a\, D(a)\, da$ is the first moment of the distribution (the average position) and is preserved with our Gaussian approximation. Note that even if the color map is linear, *the filtering problem remains non-linear* because distribution D is view-dependent. Indeed, even if the color map is linear, the resulting transfer function that we filter is the (non-linear) product of the color map and visibility. This is precisely why mipmap texture filtering yields incorrect results even in the case of linear color maps applied on surfaces.

Conclusion: Our framework is designed so as to remain accurate with surfaces whose statistics are well approximated by Gaussian distributions, and this is the case for many real-world surfaces. Even though there exist surfaces with non-Gaussian statistics, if the color map contains few high frequencies then the final distribution profile will have little impact; the final output will remain driven primarily by the average position of the distribution and the shifting effect produced by view dependence. These surfaces are also accurately filtered using our Gaussian approximation.

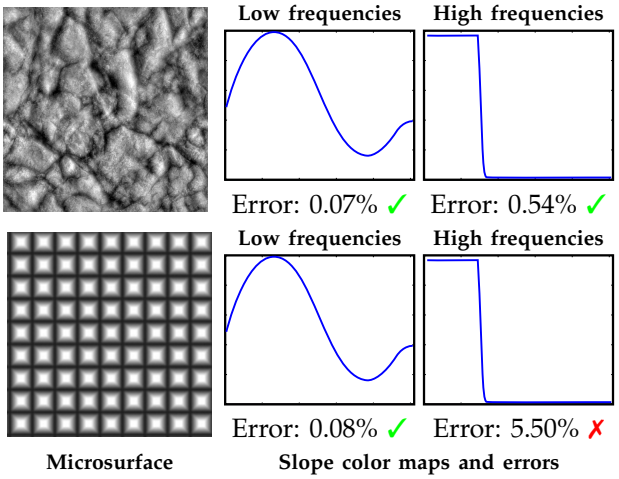


Fig. 14. Errors introduced by the Gaussian approximation in slope color mapping. The 1D color map is applied on the x component of the slope. (Top) The surface slope distribution is well approximated by a Gaussian and the error is low ($< 1\%$) for high frequencies as well as for low frequencies color maps. (Bottom) The surface slope distribution is not well approximated by a Gaussian. The error is still negligible for low frequencies color maps but becomes significative ($> 5\%$) when high frequencies are important.

9 Implementation and Results

Our method is implemented on an Intel Core i7 2.80GHz CPU with an Nvidia GTX 480. We use a prefiltered color map distribution size of 256×256 (only 192 KB of storage). The average performance of our implementation is driven by the size of the input mesh. We compare rendering performance on the Bears (74MB) and Snake (1MB) scenes by measuring the frames per second (FPS) *with* and *without* our filtering, as well as *with* and *without* tessellation.

Scene	Tess. + Filter	Filter Only	No Tess. + No Filter
Bears (Fig. 8)	85	143	145
Snake (Fig. 15)	130	800	869

Most of the rendering time is spent outside of our filtering code in the tessellation stage, and the performance difference between our filtering strategies and the naïve strategy is negligible. Our method requires only a few lines of shader code (1 for texture filtering, 3 for height-correlated filtering, and roughly 50 for orientation-correlated filtering), where we outline the most complex orientation correlation pseudocode in the supplemental material associated with this paper.

Figures 1 and 8 compare our filtering for microsurface correlated procedural color mapping examples, to ground truth (using 32×32 jittered supersampling) and to a roughly equal-time naïve filtering (i.e., $C(\int f\, dx)$) using mipmapping. Ground truth is computed offline using the GPU and the 32×32 sampling rate was chosen to resolve the

most difficult features for far away scene elements. Our results remain smooth across continuous scale transitions (see accompanying video) and, at zoom-ins (Figures 1 and 8, far right) we use adaptive tessellation to generate and display the microsurface geometry. These microsurfaces are displaced by a procedural noise function, generating Gaussian height and local-orientation statistics.

Figure 15 illustrates a linear combination of our filtering approaches. The snake's skin pattern is stored in a mipmapped displacement map whose slope statistics are approximated using a Gaussian distribution.

Texture filtering **Ground truth** **Our filtering** **Ours (close-up)**

Fig. 15. Combining techniques: the snake's skin blends a green procedural texture with a red pattern according to a procedural view-dependent function correlated to the microsurface slopes.

Our view- and light-dependent filtering clearly generates results much closer to ground truth than the only other real-time alternative (naïve color mapping of mipmapped distributions). This is because we more accurately solve the filtering integrand, including the color mapping, occlusion, and visibility terms.

Figure 16 illustrates filtering results across continuous scales.

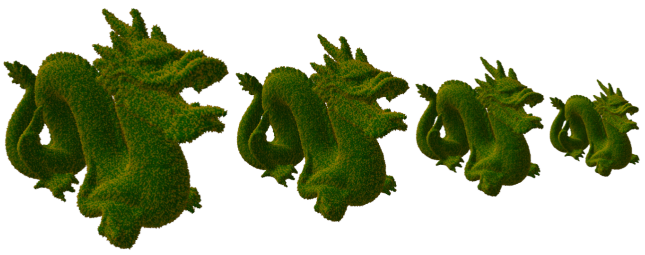


Fig. 16. View-dependent filtering across continuous scales.

10 Conclusion and Future Work

We presented a high-performance and accurate color map procedural texture filtering solution. By quickly computing filtering distributions of the procedural texture function over a pixel's weighted footprint, we are able to efficiently and accurately filter the complex non-linear behavior of color mapped textures across all scales.

Furthermore, we extend these filtering distributions to procedural microsurface color mapped details, properly modeling the correlations between the microsurface's color map, height, and local orientation. Here, filtering is view- and light-dependent, accounting for occlusion towards the viewer and masking towards the light. We finally extended our approach to transfer functions applied on surface microgeometry, thus demonstrating its generality and flexibility.

Our method is simple to integrate into existing renderers, requiring only a few lines of shader code, it has low memory and computation costs, and very closely matches ground-truth results.

Future Work: An avenue of future work would consider color maps correlated to a *joint* function of microsurface heights and slopes. In this case, a 3D filtering distribution would have to be formulated. Such a formulation may contain structure that could be exploited, e.g., for simplification via factorization, accounting for the separability of multidimensional Gaussian statistics.

Acknowledgements

The authors thank Jonathan Dupuy for several constructive discussions and Morgan Armand for his technical help. Eric Heitz was funded by the Explo'ra Doc exchange program in Rhône-Alpes (France) during his stay in Montreal. Derek Nowrouzezahrai and Pierre Poulin acknowledge financial support from NSERC and GRAND.

References

[1] K. Perlin, "An image synthesizer," in *Proceedings of ACM SIGGRAPH '85*, 1985, pp. 287–296. [Online]. Available: http://doi.acm.org/10.1145/325334.325247

[2] D. R. Peachey, "Solid texturing of complex surfaces," in *Proceedings of ACM SIGGRAPH '85*, 1985, pp. 279–286. [Online]. Available: http://doi.acm.org/10.1145/325334.325246

[3] L. Williams, "Pyramidal parametrics," in *Proceedings of ACM SIGGRAPH '83*, 1983, pp. 1–11. [Online]. Available: http://doi.acm.org/10.1145/964967.801126

[4] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. Ebert, J. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," *Computer Graphics Forum*, vol. 29, no. 8, pp. 2579–2600, 2010.

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

[6] E. Bruneton and F. Neyret, "A survey of nonlinear prefiltering methods for efficient and accurate surface shading," *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 242–260, 2012.

[7] W. Heidrich, P. Slusallek, and H.-P. Seidel, "Sampling procedural shaders using affine arithmetic," *ACM Trans. Graph.*, vol. 17, no. 3, pp. 158–176, 1998. [Online]. Available: http://doi.acm.org/10.1145/285857.285859

[8] M. Olano, B. Kuehne, and M. Simmons, "Automatic shader level of detail," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 2003, pp. 7–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=844174.844176

[9] F. Pellacini, "User-configurable automatic shader simplification," *ACM Trans. Graph. (SIGGRAPH)*, vol. 24, no. 3, pp. 445–452, 2005. [Online]. Available: http://doi.acm.org/10.1145/1073204.1073212

[10] J. Rhoades, G. Turk, A. Bell, A. State, U. Neumann, and A. Varshney, "Real-time procedural textures," in *Proceedings of ACM Symposium on Interactive 3D Graphics*, 1992, pp. 95–100.

[11] 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 *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 1999, pp. 45–53. [Online]. Available: http://doi.acm.org/10.1145/311534.311575

[12] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré, "Procedural noise using sparse Gabor convolution," in *ACM Trans. Graph. (SIGGRAPH)*, vol. 28, no. 3, 2009, pp. 54:1–10. [Online]. Available: http://doi.acm.org/10.1145/1576246.1531360

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

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

[15] M. Hadwiger, R. Sicat, J. Beyer, J. Krüger, and T. Möller, "Sparse PDF maps for non-linear multi-resolution image operations," *ACM Trans. Graph. (SIGGRAPH Asia)*, vol. 31, no. 6, pp. 133:1–12, 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366152

[16] H. Wu, J. Dorsey, and H. Rushmeier, "Characteristic point maps," *Computer Graphics Forum (EUROGRAPHICS)*, vol. 28, no. 4, pp. 1227–1236, 2009.

[17] ——, "Physically-based interactive bi-scale material design," *ACM Trans. Graph. (SIGGRAPH Asia)*, vol. 30, no. 6, pp. 145:1–10, 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024179

[18] E. Heitz and F. Neyret, "Representing appearance and pre-filtering subpixel data in sparse voxel octrees," in *Proceedings of ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, 2012, pp. 125–134.

[19] C. Bourlier, J. Saillard, and G. Berginc, "Effect of correlation between shadowing and shadowed points on the Wagner and Smith monostatic one-dimensional shadowing functions," *IEEE Trans. on Antennas and Propagation*, vol. 48, no. 3, pp. 437–446, 2000.

[20] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," *ACM Trans. Graph. (SIGGRAPH)*, vol. 21, no. 3, pp. 527–536, 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566612

[21] R. Ng, R. Ramamoorthi, and P. Hanrahan, "Triple product wavelet integrals for all-frequency relighting," *ACM Trans. Graph. (SIGGRAPH)*, vol. 23, no. 3, pp. 477–487, 2004. [Online]. Available: http://doi.acm.org/10.1145/1186562.1015749

[22] R. Ramamoorthi and P. Hanrahan, "An efficient representation for irradiance environment maps," in *Proceedings of ACM SIGGRAPH '01*, 2001, pp. 497–500. [Online]. Available: http://doi.acm.org/10.1145/383259.383317

Eric Heitz received his M.Sc. degree in Computer Science from the Karlsruhe Institute of Technology, Germany, and an Engineering degree from Ensimag, France, in 2010. He is a Ph.D. student in Computer Science at INRIA, France, under the supervision of Fabrice Neyret. His research interests include level-of-detail representations, local illumination, proceduralism, and real-time physically based rendering.

Derek Nowrouzezahrai is an Assistant Professor in the Computer Science and Operations Research department of the Université de Montréal and the co-director of UdeM's computer graphics lab. He obtained his Ph.D. and M.Sc. from the University of Toronto, both in Computer Science, and his B.A.Sc. in Computer Engineering from the University of Waterloo. Derek's has worked on realistic image synthesis techniques, including sampling approaches, real-time rendering, participating media and art-directable rendering, as well as computational geometry processing and fluid animation.

Pierre Poulin is a Full Professor in the Computer Science and Operations Research department of the Université de Montréal. He holds a Ph.D. from the University of British Columbia and a M.Sc. from the University of Toronto, both in Computer Science. He was program co-chair of Eurographics 2013, and has served on program committees of more than 50 international conferences. His research interests cover a wide range of topics, including image synthesis, image-based modeling, procedural modeling, natural phenomena, scientific visualization, and computer animation.

Fabrice Neyret is a CNRS senior researcher at INRIA Institute and LJK lab, Grenoble, France. He received a Master degree in Mathematics from Nancy University in 1989, an engineering degree from Telecom Paris in 1991, and a Ph.D. in Computer Sciences from Orsay University & INRIA Institute in 1996. He did his Postdoctoral at the University of Toronto in 1997. He also worked for various companies, including EDF (France), Thomson Digital Image (France), and Alias|Wavefront (Canada). His research interests include textures, local illumination, anti-aliasing, scalability, fluids, natural phenomena, and more generally, alternative representations and algorithms to deal with the specification, animation, and rendering of detailed large animated scenes (rivers, clouds, etc.), possibly in real-time, for games, special effects, and the understanding of macroscopic physical phenomena.