Filtering Color Mapped Textures and Surfaces

Eric Heitz[1,2] Derek Nowrouzezahrai[2] Pierre Poulin[2] Fabrice Neyret[1]
[1]INRIA-LJK (Université de Grenoble and CNRS) [2]LIGUM, Dept. I.R.O., Université de Montréal

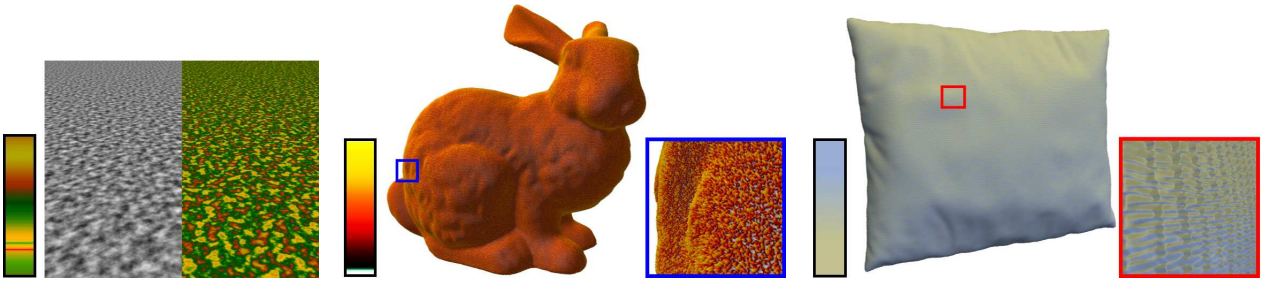


Figure 1: *An anti-aliased procedural texture produced by applying a color map to noise (left), introducing structure that cannot be generated using only the noise function. Meshes with tessellated color mapped procedural microsurface details. The color map is applied to the microsurface heights (middle)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

Abstract

Color map textures applied directly to surfaces, to geometric microsurface details, or to procedural functions (such as noise), are commonly 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 color map filtering remains an open problem for several reasons: color maps are complex non-linear functions, 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 filter color map textures applied to (macro-scale) surfaces, as well as color maps applied according to (micro-scale) 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 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.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Antialiasing; I.3.7 [Computer Graphics]: 3D Graphics and Realism—Color, shading, shadowing, and texture.

Keywords: 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 [Perlin 1985; Peachey 1985]. 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 small set of parameters.

As with standard textures, procedural textures require proper filtering to reduce aliasing, e.g., when viewed from varying distances. MIP-mapping [Williams 1983] 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. [2010] recently solve the problem of filtering procedural *noise* functions $r(x)$, computing the filtered value $\int r(x)dx$ directly from properties of r's generating process, 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 *color map* in order to obtain the final procedural texture [Musgrave 2002], 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. For our purposes, we define a *color map* $C(x)$ as a mapping of 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/MIP-mapped driving function (i.e., $C(\int f(x)\,dx)$) is not valid in general. According to recent surveys [Lagae et al. 2010; Bruneton and Neyret 2012], 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. $f = r$), as these processes have Gaussian statistics [Lagae et al. 2010], 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 micro-detail 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).

2 Previous Work

For an in-depth study of procedural noise usage and filtering, we refer readers to recent surveys [Lagae et al. 2010; Bruneton and Neyret 2012]. We instead focus on solutions to color mapped texture and geometry-correlated texture filtering. We assume that the driving function f is either tabulated or a procedural noise (i.e. $f = r$).

Filtering Color Mapped Textures: Shader-simplification [Heidrich et al. 1998; Olano et al. 2003; Pellacini 2005] 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 [Williams 1983; Rhoades et al. 1992] 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. [1999] 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. [2009] compute the spectrum of procedural noise functions, and analytically estimate the variance lost due to filtering. These methods are limited to box pre-filtering.

Worley [2002] uses an adaptive numerical integration scheme based on heuristic bounds of the spectrum of the procedurally driven color map. Better heuristics [Bergner et al. 2006] 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. [2012] 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.

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 defined 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 an abstract function f (e.g. color mapped noise, height dependent color, etc.)
$\rho(n_x, \omega_o, \omega_i)$	Bi-directional reflectance distribution function

Table 1: *The notation used throughout the paper.*

Wu et al. [2009; 2011] use characteristic point maps to find view-dependent correlations between procedural color maps and surface structures. Their method's memory usage and precomputation times precludes its application to truly dynamic textures.

Heitz and Neyret [2012] 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).

3 Preliminaries and Overview

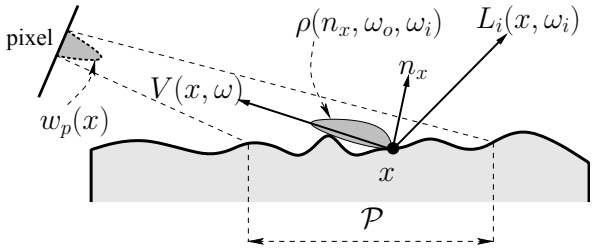


Figure 2: *The geometry of our problem. We use Gaussian statistics to filter color mapped textures and microsurface detail at all scales.*

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

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

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 [2012]. We briefly review their work

before identifying our more complete treatment of this open problem.

Existing Work on Color map Filtering: Heitz and Neyret [2012] 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

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

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

$$\Lambda(\omega) = \frac{1}{\sqrt{2\pi}}\frac{\sigma_s}{\cot\theta_i}\exp\left(-\frac{\cot^2\theta_i}{2\sigma_s^2}\right) - \frac{1}{2}\operatorname{erfc}\left(\frac{\cot\theta_i}{\sqrt{2}\sigma_s}\right)\,, \qquad (3)$$

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

$$\begin{aligned}\overline{C}(x,\omega_i,\omega_o) &= \frac{\displaystyle\int_{\mathcal{H}} C(h)\,V(h,\omega_o)\,V(h,\omega_i)\,p_h(h)\,dh}{\displaystyle\int_{\mathcal{H}} V(h,\omega_o)\,V(h,\omega_i)\,p_h(h)\,dh}\\ &= c_0 + c_1\frac{\Lambda(\omega_o)+\Lambda(\omega_i)+1}{\Lambda(\omega_o)+\Lambda(\omega_i)+2}\,, \end{aligned} \qquad (4)$$

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 variation. The latter two filtering solutions properly model the effects of occlusion towards the eye and visibility towards the light, and we outline how to combine these 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] [Bourlier et al. 2000]. 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),

$$\overline{L_i} = \int_{\mathcal{P}} L_i(x,\omega_i)\,w_P(x)\,dx \Big/ \int_{\mathcal{P}} w_P(x)\,dx\,. \qquad (5)$$

[1] Or that any correlation is sufficiently small and can be safely neglected.

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.

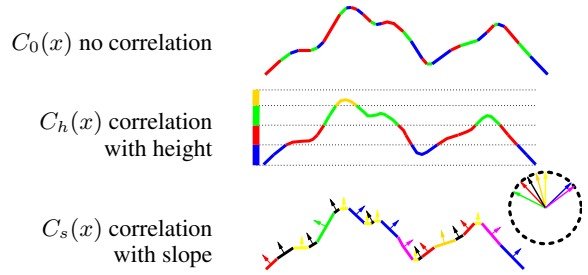


Figure 3: *We decompose the color map into three components.*

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). Note that filtering color maps of the form C_0 corresponds to the long-standing problem of filtering procedurally driven color mapped 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

$$I = \overline{L_i}\underbrace{\left[\frac{\int_{\mathcal{P}} C_0(f(x))w_P dx}{\int_{\mathcal{P}} w_P\,dx}\right]}_{\overline{C_0}}\left[\frac{\int_{\mathcal{P}} \rho(n_x)V_o(x)V_i(x)\,w_P\,dx}{\int_{\mathcal{P}} V_o(x)\,w_P\,dx}\right], \qquad (6)$$

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:

$$\overline{C_0} = \int_{\mathcal{P}} C_0(f(x))\,w_P(x)\,dx \Big/ \int_{\mathcal{P}} w_P(x)\,dx\,, \qquad (7)$$

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 (7) 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 (7) can be formulated as an inner product (expressed with angled bracket notation):

$$\overline{C_0} = \int_{-\infty}^{\infty} C_0(v)\, D_f(\mathcal{P}, v)\, dv = \langle C_0, D_f(\mathcal{P}, \cdot)\rangle \,, \qquad (8)$$

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 (8) 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 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 (8).

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, MIP-mapping 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 (8) reduces to

$$\langle C_0, D_f(\mathcal{P}, \cdot)\rangle = \sum_i \sum_j \alpha_i\, \beta_j\, \langle a_i, b_j\rangle \,, \qquad (9)$$

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 [Sloan et al. 2002; Ng et al. 2003] 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 [Bruneton and Neyret 2012] 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 [Lagae et al. 2010]. 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 below in Section 4.2 in order to devise our specific filtering solution.

4.2 Filtering 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 graphics fall into this category (see [Lagae et al. 2010]). 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 and a Gaussian with mean $\bar{f}$ is the convolution $C_0 * \mathcal{N}(0, \sigma_f^2)$ evaluated at the mean,

$$\langle C_0, \mathcal{N}(\bar{f}, \sigma_f^2)\rangle = \left[C_0 * \mathcal{N}(0, \sigma_f^2)\right](\bar{f}) \,. \qquad (10)$$

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 pre-convolutions in a 2D texture indexed by $\bar{f}$ and σ_f. At run-time, a shader efficiently computes the $\bar{f}$ and σ_f *over the footprint* $\mathcal{P}$ of the procedural (e.g. noise) function, and uses these parameters to sample $\overline{C_0}$ (see Figure 4).

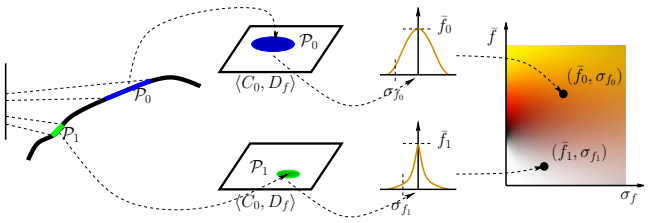


Figure 4: *We prefilter the color map applied to a procedural function (e.g. noise). We sample specially constructed filtering distributions (right) using the function's statistics over the surface patch.*

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 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

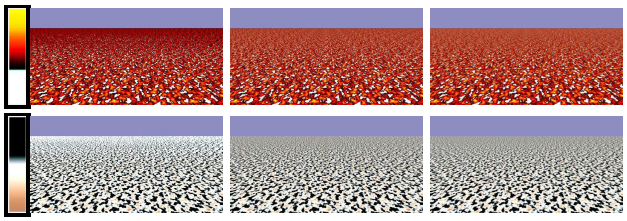
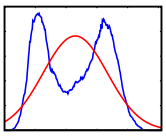


Figure 5: *Filtering color mapped textures that cannot be created using only the application of a procedural (noise) function.*

with anisotropic texture filtering methods (i.e., more complex w_P). For procedural (e.g. noise) functions f, such as Perlin noise [Perlin 1985], 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 [2009] allow for an analytical evaluation of the noise variance in the spectral domain.

We have detailed a method for computing $\overline{C_0}$, and thus Equation (6), 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 driver 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 MIP-mapping 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.



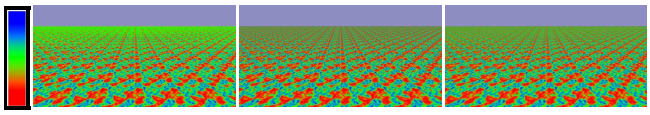


Figure 6: *The color map (left) is applied to our filtering method with a Gaussian approximation (right), which still closely matches ground truth computed with the original statistics (middle).*

5 Filtering Color Mapped Surfaces

If the color is correlated to a surface attribute, then Equation (6) 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). 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:

$$I = \overline{L_i} \underbrace{\left[\frac{\int_{\mathcal{P}} C_h(x)\, V(x,\omega_o)\, V(x,\omega_i)\, w_P\, dx}{\int_{\mathcal{P}} V(x,\omega_o)\, w_P\, dx}\right]}_{\overline{C_h}} \times \underbrace{\left[\frac{\int_{\mathcal{P}} \rho(n_x)\ \max(\cos\theta_i, 0)\ \max(\cos\theta_o, 0)\, w_P\, dx}{\int_{\mathcal{P}} \max(\cos\theta_o, 0)\, w_P\, dx}\right]}_{\overline{\rho}}, \quad (11)$$

where $\overline{\rho}$ is the filtered reflectance. We simplify the problem of solving Equation (11) – and thus, Equation (1) in this special case – to that of solving $\overline{C_h}$. Solving the reflectance filtering problem is outside the scope of our work; we employ one of the simpler techniques described in the recent survey [Bruneton and Neyret 2012].

Motivated by the flexibility of Gaussian statistics, which we already leveraged in Section 4.2, we will assume that the height distribution of our microsurface geometry is formed according to a zero-mean Gaussian process: $p_h(h) = \mathcal{N}(0, \sigma_h^2)$. In Section 7 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 (11) 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,

$$\overline{V}(\omega_o,\omega_i) = \int_{\mathcal{P}} V(x,\omega_o)\, V(x,\omega_i)\, w_P dx \Big/ \int_{\mathcal{P}} V(x,\omega_o)\, w_P dx\ , \quad (12)$$

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]:

$$\begin{aligned}\overline{V} &= \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\, P_h(h)^{\Lambda(\omega_i)}\, p_h(h)\, dh \Big/ \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\, dh \\ &= \int_{\mathcal{H}} P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\, p_h(h)\, dh \Big/ \int_{\mathcal{H}} P_h(h)^{\Lambda(\omega_o)}\, dh \\ &= \frac{1+\Lambda(\omega_o)}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}\ . \end{aligned} \quad (13)$$

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

$$\begin{aligned}\overline{C_h}(\omega_o,\omega_i) &= \frac{\int_{\mathcal{P}} C_h(x)\, V(x,\omega_o)\, V(x,\omega_i)\, w_P\, dx}{\int_{\mathcal{P}} V(x,\omega_o)\, w_P\, dx} \\ &= \overline{V}(\omega_o,\omega_i)\, \frac{\int_{\mathcal{H}} C_h(h)\, V(h,\omega_o)\, V(h,\omega_i)\, p_h(h)\, dh}{\int_{\mathcal{H}} V(h,\omega_o)\, V(h,\omega_i)\, p_h(h)\, dh} \\ &= \overline{V}(\omega_o,\omega_i) \int_{\mathcal{H}} C_h(h)\, D_h(\mathcal{P},h,\omega_o,\omega_i)\, dh \\ &= \overline{V}(\omega_o,\omega_i)\, \langle C_h, D_h(\mathcal{P},\cdot,\omega_o,\omega_i)\rangle\ , \end{aligned} \quad (14)$$

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 (8). It is important to note that this new filtering distribution depends on light and view directions.

All that remains to solve Equation (11) is an accurate and computationally efficient representation of D_h, which we present below.

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

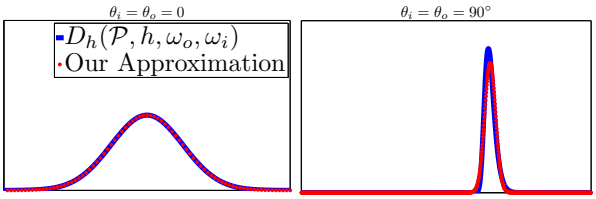


Figure 7: *Our analytic approximation of $D_h(\mathcal{P}, h, \omega_o, \omega_i)$. When $\theta_i = \theta_o = 0°$ (left) the distribution, and our approximation, match microsurface height distribution. At grazing angles, $\theta = 90°$ (right) the distribution is still well approximated by a Gaussian.*

An Efficient Height Filtering Distribution Representation: Given Equation (14) we see that

$$D_h(\mathcal{P}, h, \omega_o, \omega_i) = \frac{V(h, \omega_o)\, V(h, \omega_i)\, p_h(h)}{\int_{\mathcal{H}} V(h', \omega_o)\, V(h', \omega_i)\, p_h(h')\, dh'} . \quad (15)$$

After substituting Equation (2) (similarly to the development of Equation (13) from Equation (12), we have:

$$D_h(\mathcal{P}, h, \omega_o, \omega_i) = \frac{P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\, p_h(h)}{\int_{\mathcal{H}} P_h(h')^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\, p_h(h')\, dh'} . \quad (16)$$

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,

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

and variance

$$\begin{aligned}\sigma_d^2 &= \frac{\int_{\mathcal{H}} h^2\, P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\, p_h(h)\, dh}{\int_{\mathcal{H}} P_h(h)^{[\Lambda(\omega_o)+\Lambda(\omega_i)]}\, p_h(h)\, dh} \\ &= \left[\frac{\sigma_h}{1.0 + \alpha_\sigma\, \log\left(\beta_\sigma\, (\Lambda(\omega_o) + \Lambda(\omega_i)) + 1.0\right)}\right]^2 . \end{aligned} \quad (18)$$

We obtain the form of Equation (17) by empirically observing that: μ_d grew proportional to σ_h (leading to the multiplicative σ_h term); μ_d increased monotonically with $[\Lambda(\omega_o) + \Lambda(\omega_i)]$ with a derivative that decreased as $[\Lambda(\omega_o) + \Lambda(\omega_i)]$ increased (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 followed a similar methodology to derive the form of Equation (18).

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 height distribution. Figure 8 as well as Section 2 of the supplemental document illustrates 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 (17) and (18) will be used to sample the appropriately filtered color map value in the hierarchy at run-time[3](see Section 7).

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:

$$I = \overline{L_i} \times \overline{V} \times \overline{\rho} \times \underbrace{\left[\frac{\int_{\mathcal{P}} C_s(x)\, \max(\cos\theta_o, 0)\, w_P\, dx}{\int_{\mathcal{P}} \max(\cos\theta_o, 0)\, w_P\, dx}\right]}_{\overline{C_s}} , \quad (19)$$

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

$$\overline{C_s}(\omega_o) = \int_{\Omega_x} C(n_x)\, D_s(\mathcal{P}, n_x, \omega_o)\, dn_x = \langle C_s, D_s(\mathcal{P}, \cdot, \omega_o)\rangle , \quad (20)$$

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 (20) 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.

[3]In fact we sample with the *standard deviation* σ_d, not the variance σ_d^2.

Figure 8: *View-independent texture filtering versus our surface filtering for color mapped microsurface heights (left) and slopes (right).*

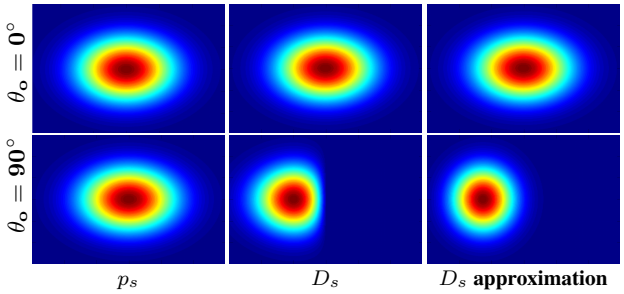


Figure 9: *(Left to right) The distribution of slopes in $\mathcal{P}$, the ground-truth view-dependent slope filtering distribution D_s, and our analytic approximation of D_s. Note that the view-dependence of D_s results in shifting and stretching of the distribution at grazing angles. Here we color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

6 Combining Techniques into Solutions

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

Linear Combinations: If the user opts to represent the final color 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 user sets the output color as a *non-linear combination* of our different correlated color mappings, then the final result 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 Implementation and Results

Our method is implemented on an Intel Core i7 2.80GHz CPU with an Nvidia GTX 480 and 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 comparing the 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. 10)	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 our supplemental document.

Figures 1 and 8 compare our filtering for microsurface correlated procedural color mapping examples, to ground truth (using 32×32 jittered super-sampling) and to a roughly equal-time naïve filtering (i.e., $C(\int f\, dx)$) using MIP-mapping. 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 finer scales (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 10 illustrates a linear combination of our filtering approaches. The snake's scaly skin pattern is stored in a MIP-mapped displacement map whose slope statistics are approximated using a Gaussian distribution.

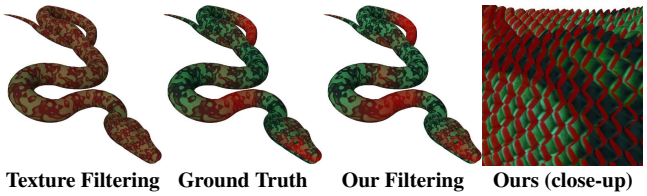


Figure 10: *Combining techniques: the snake's skin blends a green procedural texture with a red pattern according to a procedural view-dependent function correlated to the microsurface slopes.*

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 MIP-mapped distributions). This is because we more accurately model the filtering integrand, including the color mapping, occlusion, and visibility terms.

Figure 11 illustrates filtering results across continuous scales.

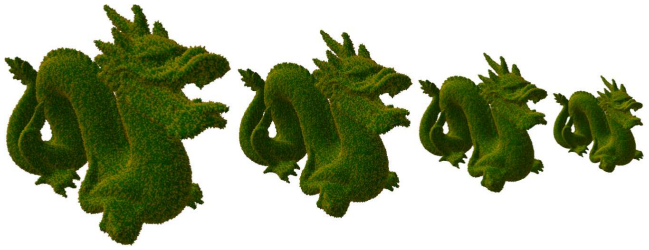


Figure 11: *View-dependent filtering across continuous scales.*

8 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.

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: Irradiance (and reflection) environment maps [Ramamoorthi and Hanrahan 2001] express unshadowed diffuse (and glossy) radiance from environment lighting as a function of surface orientation (or reflection direction). Our work can be extended and combined with these techniques to filter radiance from microsurfaces lit by environment lighting, modeling local occlusion and visibility (which, in these cases, become full spherical functions).

Another 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 multi-dimensional 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 at Montreal. Derek Nowrouzezahrai and Pierre Poulin acknowledge financial support from NSERC.

References

BERGNER, S., MOLLER, T., WEISKOPF, D., AND MURAKI, D. 2006. A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Trans. on Visualization and Computer Graphics 12*, 5, 1353 –1360.

BOURLIER, C., SAILLARD, J., AND BERGINC, G. 2000. Effect of correlation between shadowing and shadowed points on the wagner and smith monostatic one-dimensional shadowing functions. *IEEE Trans. on Antennas and Propagation 48*, 3, 437–446.

BRUNETON, E., AND NEYRET, F. 2012. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Trans. on Visualization and Computer Graphics 18*, 2, 242–260.

HADWIGER, M., SICAT, R., BEYER, J., KRÜGER, J., AND MÖLLER, T. 2012. Sparse PDF maps for non-linear multiresolution image operations. *ACM Trans. Graph. 31*, 6, 133:1–133:12.

HART, J. C., CARR, N., KAMEYA, M., TIBBITTS, S. A., AND COLEMAN, T. J. 1999. Antialiased parameterized solid texturing simplified for consumer-level hardware implementation. In *Proceedings of ACM SIGGRAPH / Eurographics Workshop on Graphics Hardware*, 45–53.

HEIDRICH, W., SLUSALLEK, P., AND SEIDEL, H.-P. 1998. Sampling procedural shaders using affine arithmetic. *ACM Trans. Graph. 17*, 3, 158–176.

HEITZ, E., AND NEYRET, F. 2012. Representing appearance and pre-filtering subpixel data in sparse voxel octrees. In *Proceedings of ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, 125–134.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse gabor convolution. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 54:1–10.

LAGAE, A., LEFEBVRE, S., COOK, R., DEROSE, T., DRETTAKIS, G., EBERT, D., LEWIS, J., PERLIN, K., AND ZWICKER, M. 2010. A survey of procedural noise functions. *Computer Graphics Forum 29*, 8, 2579–2600.

MUSGRAVE, F. K. 2002. Fractal solid textures: Some examples. In *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann, ch. 15, 447–487.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph. (SIGGRAPH) 23*, 3, 477–487.

OLANO, M., KUEHNE, B., AND SIMMONS, M. 2003. Automatic shader level of detail. In *Proceedings of the ACM SIGGRAPH / Eurographics Conference on Graphics Hardware*, 7–14.

PEACHEY, D. R. 1985. Solid texturing of complex surfaces. In *Proceedings of ACM SIGGRAPH '85*, 279–286.

PELLACINI, F. 2005. User-configurable automatic shader simplification. *ACM Trans. Graph. (SIGGRAPH) 24*, 3, 445–452.

PERLIN, K. 1985. An image synthesizer. In *Proceedings of ACM SIGGRAPH '85*, 287–296.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH '01*, 497–500.

RHOADES, J., TURK, G., BELL, A., STATE, A., NEUMANN, U., AND VARSHNEY, A. 1992. Real-time procedural textures. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, 95–100.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. (SIGGRAPH) 21*, 3, 527–536.

WILLIAMS, L. 1983. Pyramidal parametrics. In *Proceedings of ACM SIGGRAPH '83*, 1–11.

WORLEY, S. 2002. Advanced antialiasing. In *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann, ch. 5, 157–176.

WU, H., DORSEY, J., AND RUSHMEIER, H. 2009. Characteristic point maps. *Computer Graphics Forum (EUROGRAPHICS) 28*, 4, 1227–1236.

WU, H., DORSEY, J., AND RUSHMEIER, H. 2011. Physically-based interactive bi-scale material design. *ACM Trans. Graph. (SIGGRAPH Asia) 30*, 6, 145:1–145:10.