

Filtering Distributions of Normals for Shading Antialiasing

A. S. Kaplanyan^{†1} S. Hill^{‡2} A. Patney^{†1} A. Lefohn^{†1}

¹NVIDIA Research, Redmond, WA, U.S.A.

²Ubisoft Montreal, Quebec, Canada

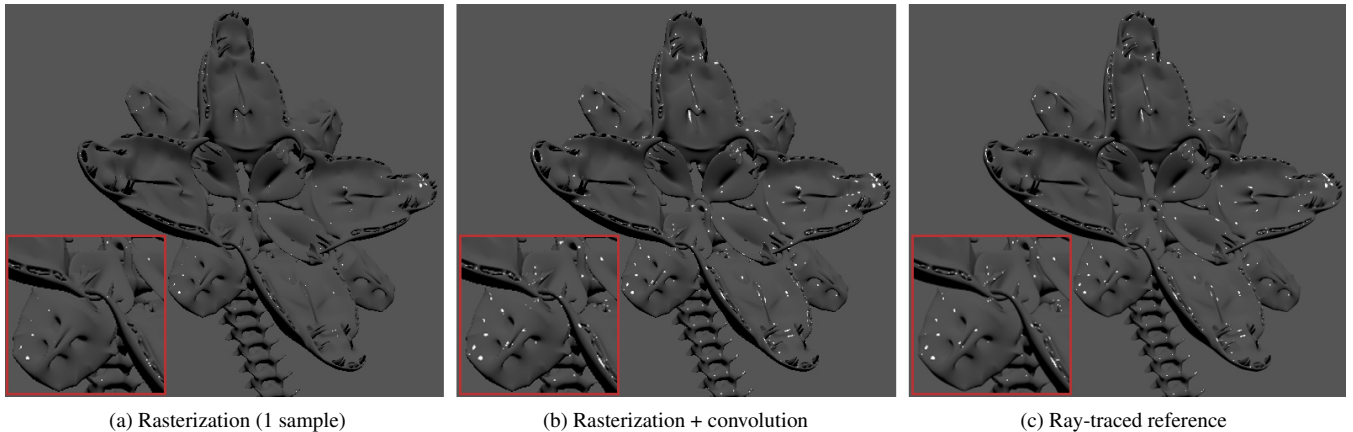


Figure 1: Prefiltering in the slope domain over the pixel footprint, with a faint diffuse term, a Beckmann BRDF with roughness $\alpha = 0.01$, and no normal maps. Left to right: (a) regular rasterization with 1 sample/pixel; (b) the same method with our analytic NDF filtering; (c) a ray-traced reference with 512 samples/pixel. Note how reliably our solution finds tiny bright highlights that usually require many samples.

Abstract

High-frequency illumination effects, such as highly glossy highlights on curved surfaces, are challenging to render in a stable manner. Such features can be much smaller than the area of a pixel and carry a high amount of energy due to high reflectance. These highlights are challenging to render in both offline rendering, where they require many samples and an outliers filter, and in real-time graphics, where they cause a significant amount of aliasing given the small budget of shading samples per pixel. In this paper, we propose a method for filtering the main source of highly glossy highlights in microfacet materials: the Normal Distribution Function (NDF). We provide a practical solution applicable for real-time rendering by employing recent advances in light transport for estimating the filtering region from various effects (such as pixel footprint) directly in the parallel-plane half-vector domain (also known as the slope domain), followed by filtering the NDF over this region. Our real-time method is GPU-friendly, temporally stable, and compatible with deferred shading, normal maps, as well as with filtering methods for normal maps.

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

1. Introduction

Modern offline and game renderers commonly use microfacet-based surface scattering models as the basis for material appearance. These models can be a challenge to correctly antialias [BN12]. A particularly important problem, and the focus of our work, is specular aliasing. Undersampling of the specular component of a bumpy surface can lead to jarring visual shimmering of bright highlights when surface features become sub-pixel [Ama92].

Surface appearance can also change under minification, which is undesirable as artistic intent should be preserved at all scales.

These issues stem from a failure to account for all sub-pixel variations of a surface. In the offline rendering world this can be addressed via super-sampling [Cro82], but this is expensive and particularly impractical for real-time applications such as games.

Partial solutions for real-time rendering are available:

- Numerous techniques handle variance from normal or displacement maps [Tok05, OB10, DHI*13], but they do not account for angular variance coming from the base surface geometry.
- Post-filtering, such as temporal antialiasing [Kar14] can reduce

[†] {akaplanyan,apatney,alefohn}@nvidia.com

[‡] stephen.hill@ubisoft.com

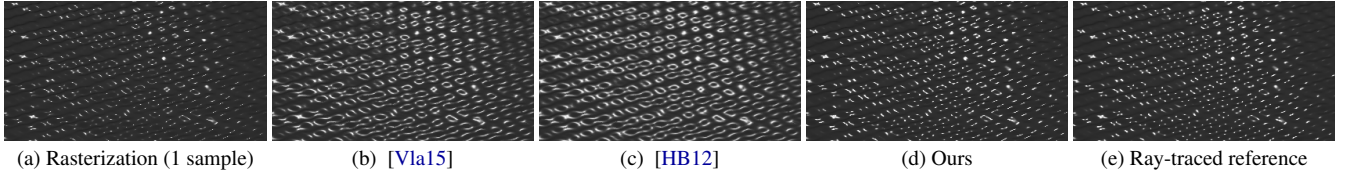


Figure 2: A tile of a wavy water surface, with GGX NDF with isotropic roughness $\alpha = 0.01$ and a faint diffuse term with a point light over it. From left to right: (a) regular rasterization with 1 shading sample/pixel; (b) specular antialiasing solution of [Vla15]; (c) specular antialiasing solution of [HB12]; (d) our solution with a rectangular NDF filtering; (e) ray-traced reference with 128 samples/pixel. Note how previous solutions change the appearance of the material.

specular shimmering, but it clamps high-intensity highlights, and it does not ensure preservation of appearance at all scales.

- Some methods attempt to address geometric variance, but they either change the appearance [HB12, Vla15] (see Fig. 2) or are restricted to special cases such as filaments [ZW07], waves [BNH10], a particular BSDF model [Ama92], or environment maps [Wil83].

In this paper, we present a new real-time method to antialias common microfacet models that accounts for sub-pixel variance in shading in a rigorous, general way. Inspired by the idea of specular bandlimiting [Ama92], our technique first identifies and transforms the filtering region (e.g., a pixel footprint) into the slope domain, where the microfacet model is defined, and then filters the principal component, the Normal Distribution Function (NDF), over this transformed region. Our approach accounts for geometric curvature, can be combined with normal map filtering techniques, and provides a connection between the variations of light path vertices and the corresponding variation induced in the NDF domain.

2. Previous Work

Modern renderers model material properties using microfacet Bidirectional Scattering Distribution Functions (BSDFs). In these models, the scattering portion of the BSDF is modeled using a distribution of microscopic specular facets, or Normal Distribution Function (NDF). Microfacet BSDFs were first introduced in physics [TS67] and later reintroduced in computer graphics [CT82] to describe light scattering during reflection off a rough surface. See Heitz [Hei14] for a recent survey on microfacet theory.

Two of the most common microfacet NDFs currently used in rendering are the Trowbridge and Reitz [TR75] model, known in graphics as GGX [WMLT07], and the Beckmann distribution [BS63]. The Beckmann NDF assumes a heightfield of microsurfaces with microfacet slopes distributed according to a bivariate Gaussian distribution in the slope domain. The GGX NDF describes the distribution of slopes produced by an ellipsoid and provides a closer match for some measured materials [WMLT07, Bur12].

The NDF of a normal-mapped or displacement-mapped surface can be arbitrarily complex, requiring careful evaluation to preserve the appearance while still avoiding aliasing. Representation of complex materials with multiple scales was studied by Westin et al. [WAT92]. Aupperle and Hanrahan [AH93] used a finite element method with adaptive subdivision of geometry under glossy

highlights. Fournier [Fou92] combined NDFs produced by multiple surfaces and bump maps with the NDF of a surface BSDF. A smooth transition between displacement maps, bump maps, and simple BSDF shading at multiple scales was proposed [BM93], using a hierarchy of multiple BSDF frequency levels as well as a modification to bump mapping. Kautz and Seidel [KS00] proposed a shift-variant BSDF model to parameterize well-known BSDF models into a non-linear basis for efficient filtering. Toksvig [Tok05] introduced a practical way of computing a variation of normals in a prefiltered mip chain of a normal map texture by fitting a Gaussian NDF. More advanced normal map filtering includes storing multiple parametric NDF lobes motivated by frequency-domain normal map filtering [HSRG07], using a mixture of BSDFs [TLQ*08], as well as linearly filterable moments of parametric distributions stored in mip chains of textures [OB10]. A similar approach was also applied to gradually convert displacement mapping into an NDF at a distance [DHI*13].

Despite progress on filtering NDFs, there is no efficient method to integrate an NDF over the region of a surface intersected by a pixel footprint, so one must resort to point sampling instead. We base our work on the seminal concept of specular antialiasing [Ama92] and address this problem by projecting an estimation of the on-surface pixel footprint into the slope domain, where microfacet NDFs are defined. With the pixel footprint extent in the slope domain, we filter the NDF across this region, enabling us to compute a higher-quality estimate of the NDF variation. More generally, our formulation integrates over perturbations caused by any of the three vertices in the eye-surface-light path.

Recently, Jakob et al. [JHY*14] and Yan et al. [YHJ*14] proposed two different methods for filtering discrete and spatially varying mesoscale NDFs. Jakob et al. [JHY*14] stochastically model the number of discrete slopes covered by the pixel footprint and can be quickly evaluated by a hierarchical subdivision of the latter in the microfacet domain. Yan et al. [YHJ*14] concurrently proposed another method for filtering spatially varying microstructure. They handle a high-resolution normal map texture by resolving a mesoscale NDF defined on a pixel footprint by hierarchically pruning irrelevant normal map texels.

Our solution employs recent progress in specular light transport. Manifold exploration (ME) [JM12] is a mutation strategy for Metropolis Light Transport [Vea98] specifically designed to improve sampling of specular and highly glossy paths. It is based on the observation that a specular vertex enforces Fermat's principle as a constraint on the half vector on a specular vertex. This work was extended to transform the integration from the on-surface area

Symbol	Description
a	Vectors are in bold
i	Unit vector of incident direction (towards the light)
o	Unit vector of outgoing direction (towards the sensor)
e	Point on sensor's aperture where the viewing ray originates
l	Point on a light source where the light ray originates
x	Point on a surface where shading is computed
n	Unit vector of shading normal at the surface point
m	Halfway unit vector on the hemisphere above x
h	Halfway vector in the slope domain
<i>M</i>	Matrix
<i>T</i>	Rotation matrix to shading tangent frame
$D(\mathbf{m})$	Normal distribution function (NDF)
\mathcal{F}	Pixel footprint projected onto a surface
\mathcal{P}	Parallelogram of ray differentials on the tangent plane

Table 1: Notation used throughout the paper.

domain to the domain of half vectors at path vertices [KHD14], and then perform the integration directly in the slope domain [HKD15]. We use the derivatives used to change the integration domain (from on-surface to slope domain) to transform the pixel footprint from its on-surface projection (e.g., obtained using ray differentials [Ige99]) into the region directly in the slope domain. Using this region, we can efficiently filter the NDF across the pixel footprint.

3. Background

The microfacet scattering distribution model assumes that scattering appears due to a microscopic geometric surface structure, which can be defined by a heightfield of microsurface heights. The full microfacet BSDF model is

$$f_s(\mathbf{i}, \mathbf{o}) = \frac{F(\mathbf{i}, \mathbf{o})D(\mathbf{m})G(\mathbf{i}, \mathbf{o})}{4|\mathbf{i} \cdot \mathbf{n}||\mathbf{o} \cdot \mathbf{n}|}, \quad (1)$$

where \mathbf{n} is the surface normal; $|\cdot|$ denotes an absolute value of a dot product, which is unified for both reflection and transmission; \mathbf{m} is a halfway unit vector between incident and outgoing directions \mathbf{i} and \mathbf{o} that defines the micronormal, D is a distribution of micronormals, called a Normal Distribution Function (NDF); and F is a Fresnel term that describes the proportion of reflected light on the interface between two media. The masking-shadowing term G accounts for self-occlusion of microfacets given directions \mathbf{i} and \mathbf{o} . See Table 1 for detailed notation.

Normal Distribution Function. The normal distribution function D at surface point \mathbf{x} is defined as a probability for a microfacet on a unit patch A to have a normal direction \mathbf{m} as

$$D(\mathbf{m}) = \frac{dA(\mathbf{m})}{A d\mathbf{m}} = \frac{dA(\mathbf{h})}{A d\mathbf{h}} \frac{d\mathbf{h}}{d\mathbf{m}} = D(\mathbf{h}), \quad (2)$$

where $A \equiv 1$ is the area of unit patch (by convention) and $dA(\mathbf{m})$ is the area of all microfacets with normals in the infinitesimal proximity around direction \mathbf{m} . We will also slightly overload the notation and, depending on the argument, will use an alternative definition of the NDF $D(\mathbf{h})$. In this case, NDF $D(\mathbf{h})$ is evaluated in the domain of slopes, with $\mathbf{h} = \mathbf{m}_{xy}/m_z$ being a projection of \mathbf{m} onto a parallel plane one unit away from the surface along the normal (see

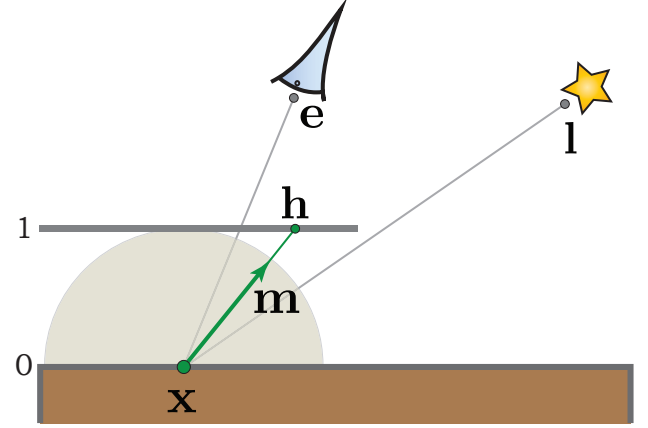


Figure 3: Notation of path vertices and half vector in various domains considered throughout the paper.

Fig. 3). Note that this change of domain requires a corresponding Jacobian, which is embedded into the $D(\mathbf{h})$ form of the NDF.

3.1. Transformation into Half-Vector Space

A transformation from the space of light path vertices to half vectors was first plotted by Chen and Arvo [CA00], explicitly formulated in manifold exploration [JM12], and further formalized in half-vector space light transport [KHD14, HKD15]. We use this transformation to estimate the region \mathcal{F} for NDF filtering in Section 4.

First, consider a direct lighting path in Fig. 3 that consists of an eye vertex \mathbf{e} , a vertex on the scene surface \mathbf{x} subject to shading, and a vertex \mathbf{l} on the light source. The halfway vector \mathbf{h} in the slope domain is then a function of all three path vertices. It can be defined in the local shading tangent frame as

$$\mathbf{h} = T(\mathbf{x}) \mathbf{h}_w, \quad (3)$$

$$\text{where } \mathbf{h}_w = \frac{\mathbf{i} + \mathbf{o}}{|\mathbf{n} \cdot (\mathbf{i} + \mathbf{o})|}$$

is the half vector in world space, formed from the incident direction $\mathbf{i}(\mathbf{x}, \mathbf{l}) = \frac{\mathbf{l} - \mathbf{x}}{\|\mathbf{l} - \mathbf{x}\|}$ and outgoing direction $\mathbf{o}(\mathbf{e}, \mathbf{x}) = \frac{\mathbf{e} - \mathbf{x}}{\|\mathbf{e} - \mathbf{x}\|}$, and $\mathbf{n}(\mathbf{x})$ is the shading normal at point \mathbf{x} . This vector is transformed by $T(\mathbf{x}) = (\mathbf{s}, \mathbf{t}, \mathbf{n})^T$, a matrix formed by the basis vectors of the orthonormal shading frame at \mathbf{x} . In order to convert the halfway vector from a hemisphere (\mathbf{m}) to a parallel plane (\mathbf{h}_w), we replace its normalization with division by the dot product with the shading normal \mathbf{n} . Note that this is the same slope domain in which a microfacet NDF is defined [HKD15].

Derivative at a Vertex. Without loss of generality, we obtain a derivative of the function \mathbf{h} with respect to the middle path vertex \mathbf{x} on the surface. It is also the derivative we will use later for filtering with the pixel footprint. Derivatives with respect to other two vertices are similarly obtained (see [Jak13] for details).

We will use the derivative matrix M (Jacobian) of $\mathbf{h}(\mathbf{x})$ with respect to \mathbf{x} to perform a change of variables to convert the domain

from vertex \mathbf{x} to half vector \mathbf{h} . The Jacobian matrix M can be computed in a straightforward manner as

$$M(\mathbf{x}) = \frac{d\mathbf{h}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial h_s}{\partial x_s} & \frac{\partial h_s}{\partial x_t} \\ \frac{\partial h_t}{\partial x_s} & \frac{\partial h_t}{\partial x_t} \end{pmatrix}, \quad (4)$$

where $*_s$ and $*_t$ are the scalar components of a vector along the corresponding vectors \mathbf{s} and \mathbf{t} of the shading tangent frame. We refer the reader to the optimized expressions for computing these derivatives in Appendix A2 of Jakob's PhD thesis [Jak13] and in half-vector space light transport [HKD15, Section 3.2]. This form can be used for ray tracing-based offline rendering methods, but we will show how to avoid the bulky computation of this full derivative when rendering with rasterization on modern GPUs, or with shading languages that provide partial derivatives, such as OSL [GSKC10].

We also use this Jacobian matrix M for a first-order Taylor approximation

$$\begin{aligned} \mathbf{h}(\mathbf{x} + \Delta\mathbf{x}) &= \mathbf{h}(\mathbf{x}) + \Delta\mathbf{x}M(\mathbf{x}) + o(\Delta\mathbf{x}), \\ \Delta\mathbf{h}(\mathbf{x}) &\approx \Delta\mathbf{x}M(\mathbf{x}). \end{aligned} \quad (5)$$

The last line is a rearrangement of the first equality, followed by a truncation of higher-order terms. In other words, we can estimate a first-order change in the half-vector domain caused by the change of vertex \mathbf{x} using this Jacobian matrix M .

Another useful form that reveals the structure of the Jacobian matrix M can be derived from Eq. 4 as

$$M(\mathbf{x}) = \mathbf{h}'(\mathbf{x}) = T(\mathbf{x})\mathbf{h}'_w(\mathbf{x}) + T'(\mathbf{x})\mathbf{h}_w(\mathbf{x}), \quad (6)$$

where \mathbf{h}'_w is a 2×2 full derivative matrix of \mathbf{h}_w ; and $T'(\mathbf{x})$ is a $2 \times 2 \times 2$ tensor derivative of the 2×2 tangent frame matrix with respect to the tangent axes \mathbf{s} and \mathbf{t} .

4. NDF Filtering for Direct Lighting

4.1. Filtering an NDF

Consider a direct lighting situation with three vertices $\mathbf{e}, \mathbf{x}, \mathbf{l}$ along the light path (see Fig. 4, left). In the general case, in order to compute the incident flux on the pixel surface, we should integrate over three dimensions: camera aperture, pixel area, and light source area. To simplify the analysis, let us assume a pinhole camera with infinitesimal aperture and a point light source with infinitesimal extent, and focus on the example of filtering an NDF with the pixel area. In other words, we assume that the vertices \mathbf{l} and \mathbf{e} stay fixed, while the vertex \mathbf{x} moves on the surface based on the outgoing direction from the pinhole camera. The light transport integral [Vea98] is then

$$I = \int_{\mathcal{F}} W(\mathbf{e}, \mathbf{x}) \mathcal{G}(\mathbf{e}, \mathbf{x}) f_s(\mathbf{e}, \mathbf{x}, \mathbf{l}) \mathcal{G}(\mathbf{l}, \mathbf{x}) L_e(\mathbf{l}, \mathbf{x}) d\mathbf{x}, \quad (7)$$

where I is the resulting incident flux onto the pixel area, \mathcal{F} is the region of the scene visible through the pixel (the *pixel footprint*), $W(\mathbf{e}, \mathbf{x})$ is the responsivity of the sensor towards the direction $\overline{\mathbf{e}\mathbf{x}}$, $\mathcal{G}(\mathbf{a}, \mathbf{b})$ is a geometric term between vertices \mathbf{a} and \mathbf{b} , $L_e(\mathbf{l}, \mathbf{x})$ is the radiance emitted from the light source towards $\mathbf{l}\mathbf{x}$, and f_s is the BSDF at point \mathbf{x} evaluated with directions $\mathbf{o} = \overline{\mathbf{x}\mathbf{e}}$ and $\mathbf{i} = \overline{\mathbf{x}\mathbf{l}}$.

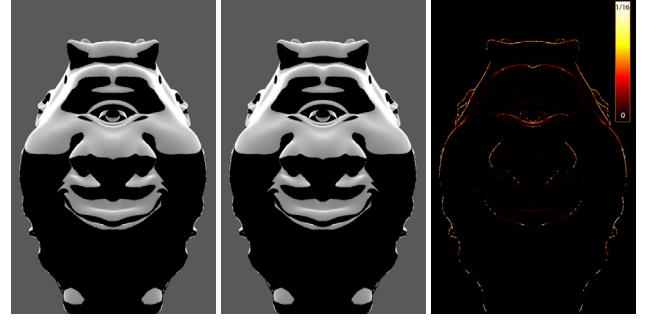


Figure 5: The OGRE scene rendered with a Beckmann BSDF that excludes the NDF itself, and a directional light with unit intensity. Left: all terms in the BSDF f_s except the NDF D are visualized and evaluated only once at the center of the pixel; middle: reference image, where these terms are supersampled with 512 samples per pixel; right: false-colored difference with HDR intensity range from 0 to 1/16 (maximum is 1).

The product $W(\mathbf{e}, \mathbf{x})\mathcal{G}(\mathbf{e}, \mathbf{x})$ is usually evaluated by sampling the point on the pixel uniformly and then finding the corresponding visible point \mathbf{x} on the surface using ray tracing or rasterization. This term is therefore perfectly importance sampled and disappears from the final estimation. The product $\mathcal{G}(\mathbf{l}, \mathbf{x})L_e(\mathbf{l}, \mathbf{x})$ is usually handled during lighting computations.

Here we apply the *far field* approximation. The far field model is based on the assumption that the distances between vertices \mathbf{e} , \mathbf{x} and \mathbf{l} are *much greater* than the largest extent of the integration region \mathcal{F} . In other words, the shading region is insignificant in size compared to the distance to the sensor and the light source. Therefore, we can assume the directions \mathbf{i} and \mathbf{o} to be constant. We can then assume that both terms $W(\mathbf{e}, \mathbf{x})\mathcal{G}(\mathbf{e}, \mathbf{x})$ and $\mathcal{G}(\mathbf{l}, \mathbf{x})L_e(\mathbf{l}, \mathbf{x})$ stay relatively constant during the integration and decorrelated from the varying terms. Therefore, we take them out of the integral (including the smoothly varying adjoining cosine terms inside the geometric terms) and integrate them separately. Denoting the integral with the product of these terms as C_p (path constant) for brevity, we obtain

$$I \approx C_p \int_{\mathcal{F}} f_s(\mathbf{e}, \mathbf{x}, \mathbf{l}) d\mathbf{x}. \quad (8)$$

This is a standard *uncorrelated shading* assumption that filters the BSDF f_s across the pixel footprint \mathcal{F} . We refer the reader to [BN12] for a thorough treatment of the subject.

Next, we insert a microfacet BSDF from Eq. 1 and apply the far field assumption again, to pull the slowly changing terms—such as the shadowing-masking term G , and a Fresnel term F , along with dot products—out of the integral. We denote the new product of the integral of these terms with the old constant C_p as C

$$I \approx C \int_{\mathcal{F}} D(\mathbf{m}(\mathbf{e}, \mathbf{x}, \mathbf{l})) d\mathbf{x} = C \int_{\mathcal{F}} D(\mathbf{h}(\mathbf{x})) d\mathbf{x}, \quad (9)$$

where $\mathbf{m}(\mathbf{e}, \mathbf{x}, \mathbf{l})$ is the halfway unit vector (in the spherical domain) that depends on all three vertices. We also rewrite the NDF with a slope-space argument $\mathbf{h}(\mathbf{x})$. Fig. 5 shows the error introduced by these approximations. Note that due to the smoothness, the integral inside C can be sparsely sampled.

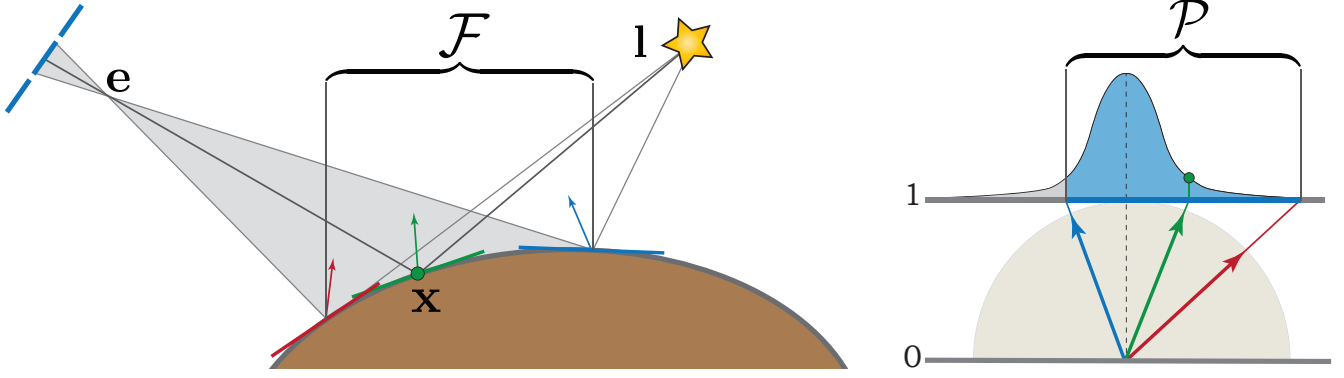


Figure 4: A flatland illustration of filtering an NDF in slope space (on the right) with the variation of half vectors induced by pixel footprint (left). A single shading point in the center of a pixel is shown with a green dot on the surface (left). On the right we also show a green dot at the NDF value corresponding to this shading point.

Next, instead of integrating the NDF over the pixel footprint, we change the integration domain from \mathbf{dx} to slope domain \mathbf{dh} and transform the pixel footprint region \mathcal{F} into the slope domain region \mathcal{P} using the corresponding linear transformation M defined earlier as $\mathcal{P} \approx M(\mathcal{F})$ (see Fig. 4). This implies that the curvature variation in the local neighborhood of the surface is insignificant. Then, the last integral becomes

$$\int_{\mathcal{F}} D(\mathbf{h}(\mathbf{x})) \mathbf{dx} \approx \int_{\mathcal{P}} D(\mathbf{h}) \left| \frac{\mathbf{dx}}{\mathbf{dh}} \right| \mathbf{dh} \approx |\mathcal{F}| \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} D(\mathbf{h}) \mathbf{dh}, \quad (10)$$

where $\mathbf{dx}/\mathbf{dh} = M$ is assumed to be constant, and we use $|\mathcal{P}| = |M||\mathcal{F}|$ by construction. This approximation of the integration region is valid under the assumption that the mapping from \mathcal{F} to \mathcal{P} is a bijection (see [KHD14] for more details).

Similar to an NDF, this integral computes an effective area of reflective microfacets, except over the *macroregion* \mathcal{F} . Note that since D does not depend on \mathbf{x} any more, the integral over the pixel footprint \mathcal{F} collapses to the area of \mathcal{F} , which is computed implicitly during sampling as previously. In other words, we substitute the NDF in Eq. 9 by its prefiltered value over the pixel footprint in the slope domain

$$\tilde{D}(\mathcal{P}) = E_{\mathcal{P}} [D(\mathbf{h})] = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} D(\mathbf{h}) \mathbf{dh}. \quad (11)$$

This integral provides an expected density of microfacets oriented within the region of slopes \mathcal{P} . This allows us to perform shading at a single point, while using the NDF that was filtered over the whole pixel footprint.

This result can be also interpreted as a convolution with a constant normalized kernel $K_{\mathcal{P}} = \mathbb{1}_{\mathcal{P}}/|\mathcal{P}|$ (with a support of \mathcal{P} shape and a value of one over the area of \mathcal{P}) and then evaluated at zero as

$$\tilde{D}(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} D(\mathbf{h}) \mathbf{dh} = \int_{\mathbb{R}^2} K_{\mathcal{P}}(\mathbf{h}) D(\mathbf{h}) \mathbf{dh} = K_{\mathcal{P}} * D. \quad (12)$$

This resulting convolution with an effective NDF kernel appears in various forms in the literature [HSRG07, BN12]. We will use it to approximate the kernel K to obtain a closed-form filtering solution for some NDFs.

4.2. Estimation of Filtering Region

In order to estimate the filtering region \mathcal{P} in the slope domain, we first estimate the pixel footprint \mathcal{F} projected onto the surface. Therefore, we begin by using a first-order approximation, ray differentials [Ige99], to project the variation of directions from the sensor's pixel onto the tangent plane of the surface vertex \mathbf{x} . This variation is then obtained as two vectors $\Delta \mathbf{x}_u$ and $\Delta \mathbf{x}_v$ on the tangent plane of the surface, corresponding to the horizontal and vertical pixel steps in the uv image plane. These two vectors form a parallelogram approximation of the pixel footprint.

We then transform each of these two differentials into half-vector space using Eq. 5 and obtain the first-order variations $\Delta \mathbf{h}_u = \Delta \mathbf{x}_u M$ and $\Delta \mathbf{h}_v = \Delta \mathbf{x}_v M$ in the slope domain, where the NDF is defined. The parallelogram \mathcal{P} formed by these vectors is then used to filter high-frequency NDFs.

4.3. Filtering Existing NDF Models

To make this method practical, we need an efficient way to compute the filtered NDF from Eq. 11. We consider two common physically based NDFs, Beckmann and Trowbridge-Reitz, and provide practical filtering solutions for them.

Beckmann NDF. In order to filter the Beckmann distribution, we first recall that it is a scaled version of a bivariate Gaussian distribution in the slope domain [BS63].

Next, we assume that the pixel reconstruction filter is a 2D Gaussian filter in image space with a standard deviation of half a pixel $\sigma_u = \sigma_v = 0.5\Delta p$. In this case, the vectors $\Delta \mathbf{h}_u$ and $\Delta \mathbf{h}_v$ represent the vectors of a standard deviation of this 2D Gaussian distribution tracked from image space into the slope domain. We then use Eq. 12 to convolve the two 2D Gaussians in the slope domain, by summing up their covariance matrices.

In order to obtain a covariance matrix from two standard deviation vectors, we use a quadratic matrix form to square the full

matrix composed of these two standard deviation vectors as

$$P = \begin{pmatrix} \Delta \mathbf{h}_u \\ \Delta \mathbf{h}_v \end{pmatrix} \begin{pmatrix} \Delta \mathbf{h}_u \\ \Delta \mathbf{h}_v \end{pmatrix}^T. \quad (13)$$

Given an anisotropic Beckmann distribution with roughness values α_s and α_t along tangent axes, its covariance matrix can be written as

$$B = \begin{pmatrix} \alpha_s^2/2 & 0 \\ 0 & \alpha_t^2/2 \end{pmatrix}, \quad (14)$$

where we use the relation $\alpha^2 = 2\sigma^2$ [BS63] between the standard deviation of a Gaussian distribution and the roughness parameter of a Beckmann distribution.

The filtered NDF is then a convolution of the Beckmann distribution with the Gaussian pixel filter projected into the slope domain. Both distributions are Gaussians, so the convolution is another 2D Gaussian with a covariance matrix B' , which is a sum of their covariance matrices

$$B' = B + P. \quad (15)$$

Note that the Beckmann distribution should then be evaluated with the full roughness matrix $2B'$ due to the scaling between the standard deviation and the roughness.

Trowbridge-Reitz NDF. The Trowbridge-Reitz/GGX NDF does not have an elegant closed-form convolution. Therefore, we use an ad hoc integration method from Eq. 11.

Moreover, we simplify the integral by circumscribing an axis-aligned bounding rectangle \mathcal{R} around the parallelogram filtering region with corresponding scalar ranges Δh_s and Δh_t along the axes \mathbf{s} and \mathbf{t} (see Fig. 6, (a)). Therefore, the integration can be decomposed as

$$D(\mathcal{P}) \approx D(\mathcal{R}) = \frac{1}{|\Delta h_s \Delta h_t|} \int_{h_s - \frac{\Delta h_s}{2}}^{h_s + \frac{\Delta h_s}{2}} \int_{h_t - \frac{\Delta h_t}{2}}^{h_t + \frac{\Delta h_t}{2}} D(s, t) ds dt, \quad (16)$$

where integration is performed over an axis-aligned rectangle in s, t and the argument \mathbf{h} for the NDF $D(\mathbf{h})$ is formed out of two scalar values (s, t) in the slope domain. See Appendix A for the full expression.

4.4. Filtering Other BSDF Terms

So far, we considered only filtering of the NDF term D . This term can take high values for smooth materials (low roughness), and is potentially unbounded, while becoming an extremely narrow peak in slope space.

In addition to the NDF, there are several other terms in the microfacet BSDF (Eq. 1) that can potentially be filtered as well. However, unlike the NDF term, all other terms, including the shadowing-masking term G and the Fresnel term F , are bounded, with values in the range $[0, 1]$. Moreover, all terms except the NDF vary smoothly and thus do not cause significant aliasing. See Fig. 5 for the variation of these terms. Therefore, we focus only on filtering the NDF term of a BSDF in this work.

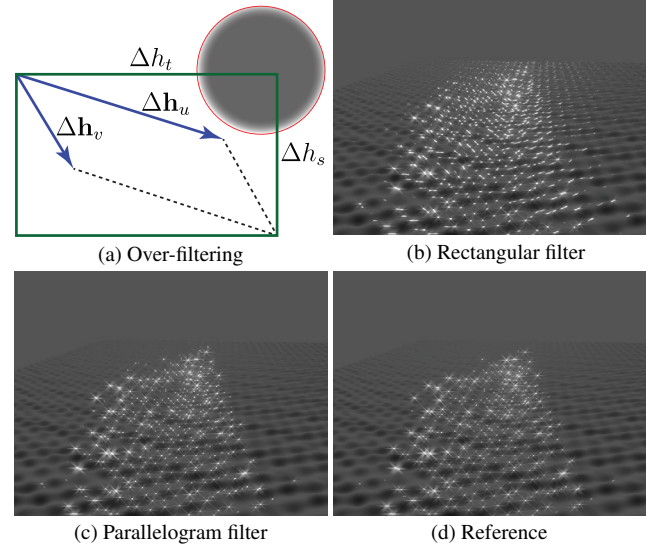


Figure 6: Water surface with Beckmann NDF ($\alpha = 0.01$). Blurred highlights occur due to rectangular filtering. Top row, left to right: (a) illustration of an over-filtering case for the rectangular filter, where the NDF peak (red circle) is not covered by the parallelogram, but is covered by its bounding rectangle; (b) filtering of the water surface with rectangular filter \mathcal{R} causes blurred highlights near the horizon. Bottom row: (c) filtering with parallelogram filter \mathcal{P} ; (d) converged reference image rendered with ray tracing with 256 samples/pixel.

5. Real-time Optimizations

In order to make NDF filtering practical for real-time use, this section describes an optimized estimation of the filtering region, approximations for making the method compatible with deferred shading [DWS*88], and a more conservative filtering of the NDF for imperfect game assets.

5.1. Efficient Computation of Jacobian on GPU

To efficiently compute the vectors $\Delta \mathbf{h}_u$ and $\Delta \mathbf{h}_v$ of the parallelogram filtering region \mathcal{P} during GPU rasterization, we do not need to perform the sophisticated machinery from [HKD15] to compute the Jacobian matrix M and ray differentials $\Delta \mathbf{x}_u$ and $\Delta \mathbf{x}_v$. Instead, we utilize the fact that shading is performed in a 2×2 quad of pixels for every shading point during GPU rasterization. Therefore, we rely on the dx/dy instructions that provide finite differencing of arbitrary variables within the same triangle on a quad. The vectors $\Delta \mathbf{h}_u$ and $\Delta \mathbf{h}_v$ are then obtained by taking the corresponding ddx and ddy derivatives of the half vector \mathbf{h} projected into the slope domain. This optimization makes the method practical for real-time graphics. Shading languages with autodifferentiation functionality, such as OSL [GSKC10], can employ the same approach.

We measure both the difference in image highlights as well as the difference in estimation of the parallelogram region \mathcal{P} between the precise computation of the Jacobian matrix M and the optimized GPU method. The difference in parallelogram is estimated as an area of non-overlapping regions between the two parallelograms.

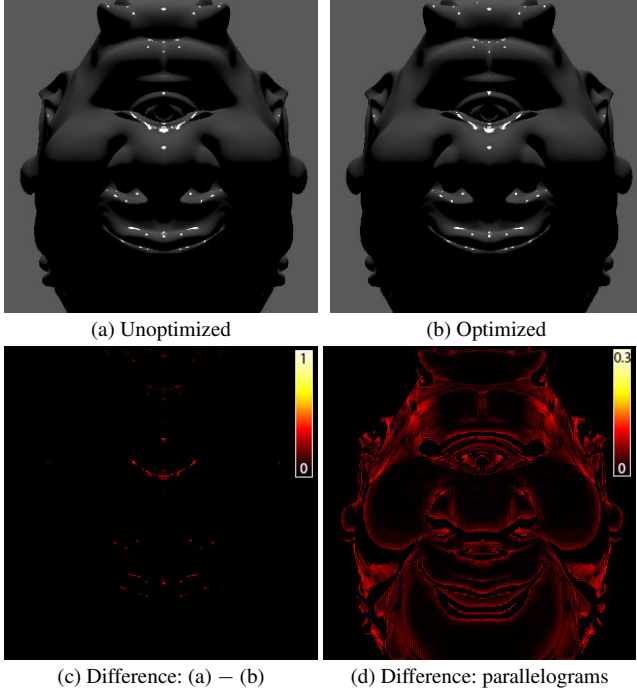


Figure 7: The OGRE scene rendered with directional light of unit intensity, a faint diffuse term and a Beckmann NDF with roughness $\alpha = 0.01$ (maximum intensity is $1/(\pi\alpha^2)$). First row: (a) derivative matrix M computed using [HKD15] in geometry and pixel shader and then multiplied by $\Delta\mathbf{x}$ computed using ddx/ddy instructions; (b) optimized computation of matrix $U = M\Delta\mathbf{x}$ using ddx/ddy finite differencing of the half vector \mathbf{h} . Second row: (c) heat map image of the intensity difference between the two images; (d) heat map image of area-based difference between the estimated parallelograms with both methods.

We provide a false-color visualization of these differences, normalized to the largest of the two areas, in Fig. 7.

5.2. Clamping the Filtering Region

The estimated filtering region \mathcal{P} is unbounded and can take large values in practice, especially at grazing viewing angles or with shading normals under the visible horizon. Since the filtering region is centered around the current half vector value, it can cover the whole slope domain, leading to false highlights in these cases. This happens due to multiple factors, such as the first-order approximation of pixel footprint, or extrapolation of quad interpolants in the case of partial pixel coverage.

On the other hand, maintaining the filtering across such large regions is not required to achieve stable antialiased NDF filtering. In practice, the region’s linear extents are far below the roughness value of 1, as demonstrated in Fig. 8. Therefore, for practical filtering, we clamp the extents of large filtering regions to a maximum of λ (we use $\lambda = 1$ in all experiments) by either rescaling the parallelogram \mathcal{P} or by simply clamping the linear values h_s and h_t for the axis-aligned rectangular region \mathcal{R} .

Further clamping from below is necessary when filtering the

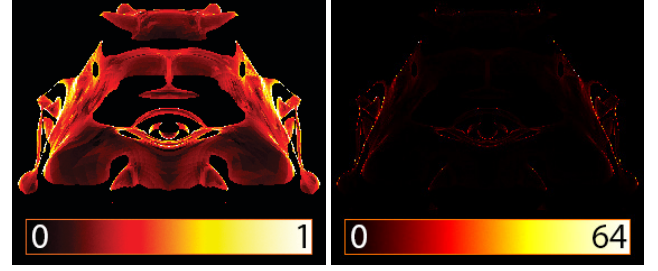


Figure 8: Heat map visualization of the maximum linear extent of an estimated filtering region \mathcal{P} in the slope domain. The magnitude of one corresponds to a filter region with maximum extent of one unit in the slope domain. Two visualizations of the same view with differently scaled ranges are provided to show both the working range of the filter size, as well as the outliers with large values at grazing angles.

GGX NDF using Eq. 16, since the division by the area of the filtering rectangle \mathcal{R} is numerically unstable when the area is small. Therefore, we clamp the sides (h_s and h_t) of the rectangle to a minimum value, $\epsilon = 10^{-3}$.

5.3. Stable Biased Filtering

In practice, a tight filtering region may not guarantee temporal stability due to various issues with production geometry, such as many small submanifolds, non-manifold geometry, and imprecise or improperly smoothed shading normals. The filtering region can be either tight and more prone to aliasing, or conservatively large and thereby over-filter, leading to excessive blurring of the highlight. This is a well-known noise-bias trade-off that has been thoroughly analyzed in the context of texture filtering (e.g., [Hec89]). Even though NDF filtering is performed in a different domain, the trade-off is the same: we either (1) apply a tight filtering region and therefore rely on the quality of the input geometry and shading normals; or (2) conservatively estimate the filtering region and therefore accept slightly blurred specular highlights. Option 1, while being prone to aliasing, provides more accurate highlight shapes. Option 2, on the other hand, while over-blurring the highlights, provides more temporal stability and more robust antialiasing.

Though it appears to be a regular trade-off, with both options having their own advantages and disadvantages, the second option is generally favored for real-time graphics [Hec89]. For example, all GPU-accelerated texture filtering methods perform a conservative estimation of pixel footprint in order to reduce aliasing.

Following this same reasoning, we propose a more conservative estimation of the filtering region to provide a robust solution for antialiasing and temporal stability. At the cost of slightly over-blurring specular highlights, with this approach we achieve a robust and temporally stable filtering solution, especially for imperfect geometry and shading normals.

Instead of using a tight first-order region of a parallelogram \mathcal{P} , for conservative filtering we use a slope-domain axis-aligned rectangle \mathcal{R} for all NDFs. This rectangle \mathcal{R} is obtained from the parallelogram \mathcal{P} by computing the largest linear extent along each axis

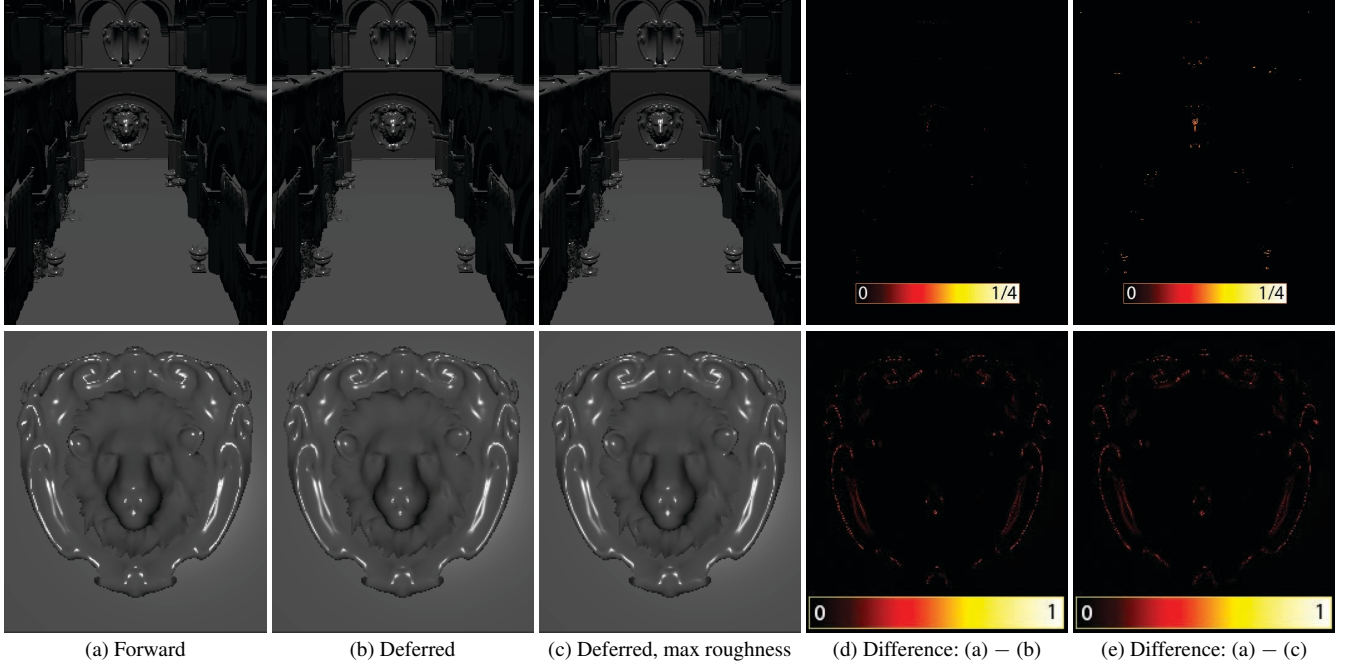


Figure 9: Deferred shading with NDF filtering, for Beckmann (top row) and GGX (bottom row) NDFs (both with roughness $\alpha = 0.01$), with two options for storing prefiltered roughness in G-Buffer. From left to right: (a) forward rendering with no MSAA and NDF filtering; (b) deferred shading with prefiltered NDF and with two G-Buffer channels for storing the anisotropic roughness; (c) same as (b), but storing the maximum roughness in a single channel; (d) HDR intensity difference image between (a) and (b); (e) difference image between (a) and (c).

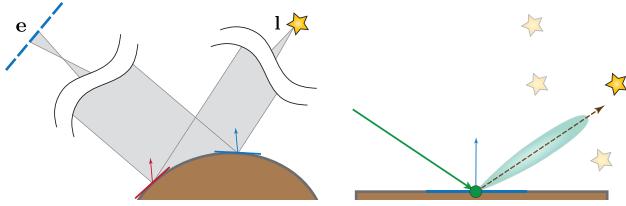


Figure 10: Assumptions for deferred shading. Left: the far field model assumes that both the light source and camera are far enough away to neglect changes in incident and outgoing directions [Ama92]. Right: we assume that the direction to the light is close to the perfect reflected viewing direction in order to conservatively estimate the worst-case change in half vector.

of the slope domain. We set the axes to be the tangent vectors \mathbf{s} and \mathbf{t} (Fig. 6, (a)).

Filtering over the axis-aligned rectangle \mathcal{R} potentially leads to a slight over-blurring of specular highlights, as demonstrated in Fig. 6, while providing stable alias-free results in all of our tested scenes. It is also the only available option we provide for the GGX NDF, though more accurate and efficient filtering of this distribution remains an open question.

5.4. Combining with Deferred Shading

Deferred shading [DWS*88] is a widespread rendering method in modern real-time applications such as games. Rendering is split into two parts: the first pass renders a scene and outputs shading

attributes into a G-Buffer; one or more subsequent shading passes are then performed in image space in order to shade the visible shading points with multiple light sources. The challenge of NDF filtering with deferred shading is that we cannot determine the half vector \mathbf{h} during the G-Buffer generation pass, since the direction to the light source \mathbf{i} is unknown. On the other hand, after this pass the Jacobian matrix cannot be computed because we do not have access to the triangle with its shading normals and their derivatives.

Recall that the Jacobian matrix M can be written using a chain rule as shown in Eq. 6. For deferred shading, we apply a far field assumption (Fig. 10, left) and thus let $\mathbf{h}'_w(\mathbf{x}) \approx 0$, obtaining

$$M(\mathbf{x}) \approx T'(\mathbf{x})\mathbf{h}_w(\mathbf{x}). \quad (17)$$

We can compute the tensor derivative of the shading frame during the G-Buffer generation pass, but the value of $\mathbf{h}_w(\mathbf{x})$ remains unknown.

Another assumption is that filtering is only required for highly glossy materials (Fig. 10, right), where we can assume that $\mathbf{h}_w(\mathbf{x}) \approx \mathbf{n}_w(\mathbf{x})$. In order to avoid degenerate (zero) derivatives with this approximation, we first compute an average normal within the shading quad and then set all \mathbf{h}_w within the quad to this value. Taking both assumptions into account, we compute an approximation of the filtering region vectors $\Delta\mathbf{h}_u$ and $\Delta\mathbf{h}_v$, and filter the NDF (as discussed above) during the G-Buffer generation pass, before storing the final filtered roughness.

NDF Prefiltering with Compact Storage. For the Beckmann NDF we simply perform a convolution using Eq. 15. Note that be-

cause we use a rectangular filtering region \mathcal{R} , the covariance matrix of the corresponding Gaussian is a diagonal matrix, therefore the resulting matrix B' is also a diagonal matrix and thus can be stored as a new prefiltered anisotropic roughness in the G-Buffer. In cases where the G-Buffer only has one channel available for storing roughness, we take the maximum of the two roughness values along \mathbf{s} and \mathbf{t} . This way we conservatively store the largest filter extent. See Fig. 9, top row for comparisons of storage options for the Beckmann NDF.

Approximate Prefiltering for Trowbridge-Reitz NDF. In the case of the GGX (Trowbridge-Reitz) NDF, we would like to store the same prefiltered roughness. However, the rectangular integral in Eq. 16 requires both roughness values and the rectangular filtering region dimensions to be stored in the G-Buffer for subsequent shading. This requires three to four G-Buffer channels (depending on the storage options for roughness), which is an undesirable memory and bandwidth burden. To avoid this, we instead store an *effective roughness* α'_s and α'_t , which is a result of approximating the convolution by fitting the filtered distribution back to a new GGX NDF with these roughness parameters.

In order to fit the effective roughness, we evaluate the closed-form convolution $K_{\mathcal{R}} * D$ of the rectangular filtering region with GGX NDF at $\mathbf{h} = 0$, by taking the integral

$$\begin{aligned} \mathcal{C}(\alpha_s, \alpha_t, \Delta h_s, \Delta h_t) &= K_{\mathcal{R}} * D(0) \\ &= \int_{\mathcal{R}^2} D(s, t) K_{\mathcal{R}}(s, t) ds dt, \end{aligned} \quad (18)$$

where the integration happens over the whole slope domain and the result is the function of GGX roughness values α_s and α_t , as well as the sides of the rectangular region Δh_s and Δh_t . This function is no longer a GGX NDF and has the peak value at the origin after convolving with the rectangular kernel. In order to fit it back to another GGX distribution with effective roughness values α'_s and α'_t , we first equate the peak value of the fitted GGX to the peak values of the convolution

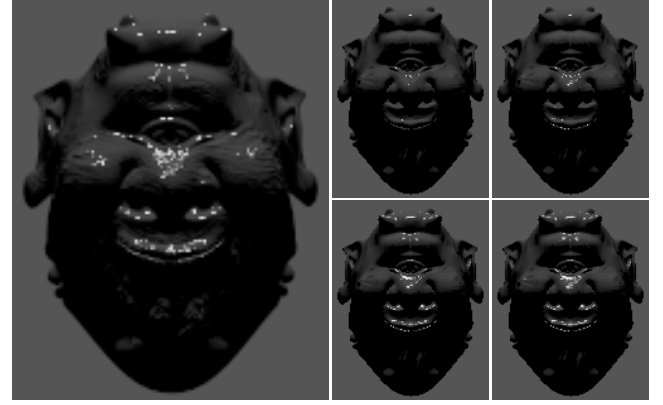
$$D(\alpha'_s, \alpha'_t)(0) = \frac{1}{\pi \alpha'_s \alpha'_t} \approx \mathcal{C}(\alpha_s, \alpha_t, \Delta h_s, \Delta h_t), \quad (19)$$

where $D(\alpha'_s, \alpha'_t)(0)$ is the fitted GGX NDF with effective roughness parameters evaluated at zero. Unfortunately, this equation is under-constrained if we were to solve it with respect to α'_s and α'_t .

Therefore, we assume the following approximations $\alpha'_s \equiv \alpha'_t \equiv \alpha'$, $\alpha_s \equiv \alpha_t \equiv \alpha$, and $\Delta h_s \equiv \Delta h_t \equiv \Delta h$ and solve Eq. 19 with these assumptions. The resulting solution

$$\begin{aligned} \alpha &\approx \frac{\sqrt{\Delta h} \sqrt{4\alpha^2 + \Delta h^2}}{2 \sqrt{\tan^{-1} \left(\frac{\Delta h}{\sqrt{4\alpha^2 + \Delta h^2}} \right)}} \\ &\approx \sqrt{\alpha^2 + \Delta h^2} \end{aligned} \quad (20)$$

is then evaluated for each triplet of parameters along the \mathbf{s} and \mathbf{t} axes separately. The last approximation is another fit done manually with a simpler function, which fits well when the argument values are low. In practice it is indistinguishable from the first approximation, so we use this simpler form for the results. Because this approximation is identical to the filtering of the Beckmann BSDF,



(a) Reference (b) Normal map + NDF filtering (c) LEAN map + NDF filtering

Figure 11: The OGRE scene rendered with normal mapping and LEAN mapping, with a faint diffuse term and a GGX NDF with roughness $\alpha = 0.01$. From left to right: (a) reference image with 1024 samples/pixel; (b) top: regular normal mapping (hereafter, 1 sample/pixel), bottom: regular normal mapping with NDF filtering; (c) top: LEAN mapping, bottom: LEAN mapping with NDF filtering. Advanced methods better reconstruct specular highlights.

we call it a Beckmann proxy for GGX, or simply *GGX Proxy*. See Fig. 9, bottom for the various roughness storage options for the GGX NDF and their associated accuracy.

5.5. Combining with Normal Map Filtering

We can combine our technique with both normal mapping and filtering techniques for normal maps. The combination with normal maps is achieved by rotating the shading frame according to the normal from a normal map, before computing the derivative of the shading frame. This is handled automatically on the GPU via the dx/dy derivatives after applying regular normal mapping to the shading tangent frame.

Our method is orthogonal to normal map filtering methods. One of them, Linear Efficient Antialiased Normal (LEAN) Mapping [OB10], stores and filters (over a mip chain) the first and second-order moments of the slope-domain normals. These terms are then fetched at runtime to estimate the parameters of a bivariate Gaussian (Beckmann) distribution.

To combine this filtering method with our NDF filtering, we do the following. We first interpret the first-order moments as a regular shading normal coming from the normal map and perturb the shading tangent frame with it. We then compute a covariance matrix Σ of the estimated Beckmann distribution (Eq. 5 in [OB10]) and interpret this as another kernel filter $K_{\mathcal{L}}$ induced by variations of half vectors caused by the normal map.

Finally, using the convolution formulation of NDF filtering from Eq. 12, we perform a nested convolution

$$\tilde{D} = K_{\mathcal{L}} * K_{\mathcal{P}} * D = (K_{\mathcal{L}} * K_{\mathcal{P}}) * D = K' * D. \quad (21)$$

To filter the Beckmann NDF, it becomes a trivial summation of

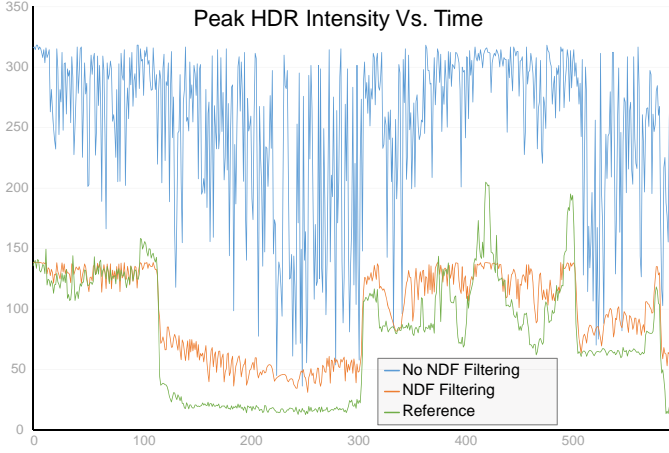


Figure 12: Temporal stability of GGX NDF filtering of the first sequence in the accompanying video with the OGRE scene. This plot compares variation of the peak High Dynamic Range (HDR) image intensity of specular highlights with (orange plot) and without (blue plot) NDF filtering, as well as the reference ray-traced solution with 512 sample/pixel (green plot). We plot frame count on the x axis, against the maximum linear HDR intensity (unitless) on the y axis.

covariance matrices (as with Eq. 15). To filter the GGX NDF, we compute the axis-aligned bounding rectangle around the vectors of the standard deviations, by using the approximate Eq. 20. Results of this combination are demonstrated in Fig. 11.

6. Results

We evaluate NDF filtering using an OpenGL-based real-time GPU renderer and shaders written in GL Shading Language (GLSL). Our renderer supports both forward and deferred rendering modes. For performance measurements, we used a workstation with an NVIDIA Quadro M6000 GPU. We use low values for the roughness parameter (usually ≤ 0.01) in all of our experiments, as with high roughness the filtering effect is both less prominent and less essential.

6.1. Image Quality

The primary advantage of NDF filtering is its ability to produce images that are closer to the ground truth. As we can see in Fig. 1, using NDF filtering at just 1 sample per pixel is similar in quality to ray tracing with 512 samples per pixel.

Fig. 9 compares image quality using our GGX filtering approximation in a deferred renderer. The difference in quality between NDF filtering in forward rendering and approximate deferred rendering implementation shows that our approximations maintain the quality of NDF filtering for both forward and deferred renderers.

Fig. 11 shows the image quality of NDF filtering integrated with normal mapping and LEAN mapping. In both cases, the quality of the final image is superior to images generated without the use of NDF filtering, and is closer to the ground truth. NDF filtering complements normal mapping and prefiltered normal mapping techniques in these cases.





Scene	Normals	Forward			Deferred	
		B	G	G _p	B	G
	Vertex	0.02	0.14	0.03	0.06	0.09
	Map	0.02	0.15	0.03	0.07	0.08
	LEAN	0.02	0.13	0.02	0.06	0.06
	Vertex	0.09	0.53	0.18	0.19	0.28
	Vertex	0.32	2.40	0.49	0.46	0.78
	Map	0.64	2.74	0.53	0.49	0.93
	LEAN	0.29	1.50	0.22	0.45	0.68
	Vertex	0.32	3.39	0.52	0.62	1.13
	Map	0.63	3.73	0.60	0.78	1.29
	LEAN	0.48	3.36	0.33	0.88	1.02

Table 2: Performance overhead of NDF filtering, measured as the change in execution time of shaders due to addition of NDF filtering. All timings are in milliseconds and correspond to images rendered at 3840×2160 pixels with 1 sample per pixel on an NVIDIA Quadro M6000 GPU. B represents the Beckmann BSDF implementation, G represents the GGX BSDF, and G_p represents the GGX Proxy distribution.

Temporal Stability. Fig. 12 shows a comparison of temporal stability in animated scenes with and without NDF filtering to the reference stability. In order to perform this measurement, we rendered High Dynamic Range (HDR) images of the first and second sequences of the accompanying video (generated with and without the use of NDF filtering) as well as the ray-traced sequence with converged highlights. Since flickering is most objectionable on points of high-intensity illumination, we track the variation in the maximum intensity across the image (L_∞ norm) with frame progression. NDF filtering generates images that are more stable and closer to the reference peak intensity than those without its use. This is a conservative metric, therefore the regions where blue and orange plots are close correspond to fortunate periods where the non-filtered shading is stable. On the other hand, there are under- and over-filtering regions in our approach (compared to the reference) due to the first-order approximation and the rectangular GGX filter region.

6.2. Performance

We measure the impact of NDF filtering on the performance of existing shading systems. We investigate this impact from two perspectives: change in instruction count of shaders modified to perform NDF filtering, and influence on the time taken by shaders.

Table 2 shows the difference in milliseconds of per-frame GPU execution time using conventional shading techniques, and those using NDF filtering. We compare performance across multiple axes: different scenes of varying complexity, forward as well as deferred rendering modes, shading normals obtained using vertex normals, normal maps, and LEAN mapping, and BSDFs including Beckmann, GGX, and GGX Proxy. In almost all cases, the cost of additional computation for NDF filtering is negligible. The only exceptions are shaders that filter GGX BSDFs (around 4 ms for San Miguel for a shader that takes 12 ms). If accurate GGX evaluation

Renderer	Forward			Deferred		
	Vertex	Map	LEAN	Vertex	Map	LEAN
Beckmann	19	19	11	24	25	36
GGX	272	272	267	—	—	—
GGX Proxy	22	22	17	24	25	36

Table 3: Instruction overhead of using our technique. We used `glslc` [Kub13] to obtain instruction counts of fragment shaders with and without NDF filtering, and report the difference above. The impact of NDF filtering in most scenarios is between 10 and 40 static instructions. Using the GGX BSDF in a forward renderer adds more than 200 instructions. The approximate GGX Proxy evaluation is a lightweight alternative.

is expensive for an application, our approximate GGX Proxy evaluation serves as a lightweight alternative with reasonable image quality (see Fig. 9).

Table 3 shows the impact of NDF filtering measured as the number of instructions added by modifying the above shading configurations to incorporate NDF filtering. We measured instruction counts by using the open-source tool `glslc` [Kub13], which outputs pseudo-assembly for GLSL shaders compiled for NVIDIA GPUs. Our measurements show that except for shaders with full GGX filtering, NDF filtering only adds 10–40 static instructions across all shader configurations, which is acceptable in most applications. As before, full GGX filtering is heavier, as it adds 250–300 instructions, whereas the alternative GGX Proxy remains lightweight without significant quality degradation.

Limitations. We address aliasing that comes from specular shading. Other sources of aliasing such as visibility are not covered, therefore no improvements should be expected with respect to geometric aliasing. We only take a single quadric into account when filtering an NDF. On geometry with high-frequency details, submanifolds can change rapidly, causing geometric aliasing during occlusion or disocclusion. Since the filtering method works on shading normals, it is crucial to have accurate shading normals on geometry, as they define the virtual shading surface used for NDF filtering.

7. Conclusions

In this work, we provide a framework for filtering microfacet NDFs caused by variations along the light path dimensions (e.g., caused by a finite pixel footprint). These variations can be efficiently transformed into the slope domain, over which we can integrate the NDF. This allows us to estimate the NDF response across the whole pixel footprint by doing the evaluation at a single shading sample. We can also find small highlights and, more importantly, their average intensity across the pixel. The method shows good real-time performance, as well as temporal stability, while being compatible with other real-time methods, such as deferred shading and normal map prefiltering.

8. Acknowledgments

We would like to thank Eric Heitz, Naty Hoffman, Eric Enderton, Chris Wyman, Tim Foley, Cyril Crassin, and Louis Bavoil for helpful discussions; Nir Benty and Marco Salvi for helping with the

rendering infrastructure, Christoph Kubisch for the `GLSLC` tool, as well as the anonymous reviewers for their valuable feedback. Thanks to Keenan Crane for the Desert Rose model in the teaser and the Ogre model (available at <http://www.cs.cmu.edu/~kmc Crane/Projects/ModelRepository/>), Samuli Laine and Tero Karras for the hairball model, Frank Meinel and Efgeni Bischoff for remodeling Crytek Sponza, Wenzel Jakob for Gane-sha model, and Guillermo M. Leal Llaguno for San Miguel model.

References

- [AH93] AUPPERLE L., HANRAHAN P.: A hierarchical illumination algorithm for surfaces with glossy reflection. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH, pp. 155–162. 2
- [Ama92] AMANATIDES J.: Algorithms for the detection and elimination of specular aliasing. In *Proceedings of the Conference on Graphics Interface* (1992), pp. 86–93. 1, 2, 8
- [BM93] BECKER B. G., MAX N. L.: Smooth transitions between bump rendering algorithms. In *Proceedings of Computer graphics and interactive techniques* (1993), SIGGRAPH, pp. 183–190. 2
- [BN12] BRUNETON E., NEYRET F.: A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics* 18, 2 (2012), 242–260. 1, 4, 5
- [BNH10] BRUNETON E., NEYRET F., HOLZSCHUCH N.: Real-time Realistic Ocean Lighting using Seamless Transitions from Geometry to BRDF. *Computer Graphics Forum* 29, 2 (2010), 487–496. 2
- [BS63] BECKMANN P., SPIZZICHINO A.: *The scattering of electromagnetic waves from rough surfaces*. International series of monographs on electromagnetic waves. Pergamon Press, 1963. 2, 5, 6
- [Bur12] BURLEY B.: Physically-based shading at Disney. *ACM SIGGRAPH Courses* (2012), 10:1–10:7. 2
- [CA00] CHEN M., ARVO J.: Theory and application of specular path perturbation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 19, 4 (2000), 246–278. 3
- [Cro82] CROW F. C.: Computational issues in rendering anti-aliased detail. In *COMPCON, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference* (1982), pp. 238–244. 1
- [CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *Computer Graphics (Proc. SIGGRAPH)* 1, 1 (1982), 7–24. 2
- [DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32, 6 (2013), 211:1–211:11. 1, 2
- [DWS*88] DEERING M., WINNER S., SCHEDIWIY B., DUFFY C., HUNT N.: The triangle processor and normal vector shader: A VLSI system for high performance graphics. *Computer Graphics (Proc. SIGGRAPH)* 22, 4 (1988), 21–30. 6, 8
- [Fou92] FOURNIER A.: Normal distribution functions and multiple surfaces. In *Graphics Interface Workshop on Local Illumination* (1992), pp. 45–52. 2
- [GSKC10] GRITZ L., STEIN C., KULLA C., CONTY A.: Open shading language. In *ACM SIGGRAPH Talks* (2010), ACM Transactions on Graphics (Proc. SIGGRAPH), pp. 33:1–33:1. 4, 6
- [HB12] HILL S., BAKER D.: Rock-solid shading: Image stability without sacrificing detail. In *Advances in Real Time Rendering*, Tatarchuk N., (Ed.). ACM SIGGRAPH Courses, 2012. 2
- [Hec89] HECKBERT P. S.: *Fundamentals of Texture Mapping and Image Warping*. Tech. Rep. UCB/CSD-89-516, EECS Department, University of California, Berkeley, 1989. 7

- [Hei14] HEITZ E.: Understanding the masking-shadowing function in microfacet-based BRDFs. *Journal of Computer Graphics Techniques (JCGT)* 3, 2 (2014), 48–107. 2
- [HKD15] HANIKA J., KAPLANYAN A. S., DACHSBACHER C.: Improved half vector space light transport. *Proc. Eurographics Symposium on Rendering* 34, 4 (2015), 65–74. 3, 4, 6, 7
- [HSRG07] HAN C., SUN B., RAMAMOORTHY R., GRINSFELD E.: Frequency domain normal map filtering. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26, 3 (2007). 2, 5
- [Ige99] IGEHY H.: Tracing ray differentials. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (1999), SIGGRAPH, pp. 179–186. 3, 5
- [Jak13] JAKOB W.: *Light transport on path-space manifolds*. PhD thesis, Cornell University, 2013. 3, 4
- [JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHY R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014), 115:1–115:10. 2
- [JM12] JAKOB W., MARSCHNER S.: Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 58:1–58:13. 2, 3
- [Kar14] KARIS B.: High-quality temporal supersampling. In *Advances in Real-Time Rendering in Games, SIGGRAPH Courses*. (2014). 1
- [KHD14] KAPLANYAN A. S., HANIKA J., DACHSBACHER C.: The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014). 3, 5
- [KS00] KAUTZ J., SEIDEL H.-P.: Towards interactive bump mapping with anisotropic shift-variant BRDFs. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2000), pp. 51–58. 2
- [Kub13] KUBISCH C.: glslc: Simple glsl compilation checker (uses display driver), 2013. <https://github.com/pixeljetstream/glslc>. 11
- [OB10] OLANO M., BAKER D.: LEAN mapping. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2010), pp. 181–188. 1, 2, 9
- [TLQ*08] TAN P., LIN S., QUAN L., GUO B., SHUM H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics* 14, 2 (2008), 412–425. 2
- [Tok05] TOKSVIG M.: Mipmapping normal maps. *Journal of Graphics, GPU, and Game Tools* 10, 3 (2005), 65–71. 1, 2
- [TR75] TROWBRIDGE T. S., REITZ K. P.: Average irregularity representation of a rough surface for ray reflection. *Journal of the Optical Society of America* 65, 5 (1975), 531–536. 2
- [TS67] TORRANCE K. E., SPARROW E. M.: Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America* 57, 9 (1967), 1105–1112. 2
- [Vea98] VEACH E.: *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1998. AAI9837162. 2, 4
- [Vla15] VLACHOS A.: Advanced VR rendering, 04 2015. Game Developers Conference Talk. 2
- [WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. *Computer Graphics (Proc. SIGGRAPH)* 26, 2 (1992), 255–264. 2
- [Wil83] WILLIAMS L.: Pyramidal parametrization. *Computer Graphics (Proc. SIGGRAPH)* 17, 3 (1983), 1–11. 2
- [WMLT07] WALTER B., MARSCHNER S., LI H., TORRANCE K.: Microfacet models for refraction through rough surfaces. In *Proc. Eurographics Symposium on Rendering* (2007), pp. 195–206. 2
- [YHJ*14] YAN L.-Q., HAŠAN M., JAKOB W., LAWRENCE J., MARSCHNER S., RAMAMOORTHY R.: Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014), 116:1–116:9. 2
- [ZW07] ZINKE A., WEBER A.: Light scattering from filaments. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (2007), 342–356. 2

Appendix A: Filtering Trowbridge-Reitz NDF

Here we provide a full expression for the integral in Eq. 16 as

$$\text{Eq. (16)} = \frac{1}{2\pi|\Delta h_s \Delta h_t|} \left(\begin{aligned} &\varphi(h_s^0, h_t^0, \alpha, \beta) - \varphi(h_s^0, h_t^1, \alpha, \beta) + \varphi(h_s^1, h_t^1, \alpha, \beta) - \varphi(h_s^1, h_t^0, \alpha, \beta) + \\ &\varphi(h_t^0, h_s^0, \beta, \alpha) - \varphi(h_t^0, h_s^1, \beta, \alpha) + \varphi(h_t^1, h_s^1, \beta, \alpha) - \varphi(h_t^1, h_s^0, \beta, \alpha) \end{aligned} \right),$$

where h_s and h_t are coordinates of the current half vector used for shading (computed at the center of the pixel) in the local shading frame projected onto the parallel plane. Integration is performed over a rectangular filtering region that spans from $h_s^0 \equiv h_s - \Delta h_s$ to $h_s^1 \equiv h_s + \Delta h_s$ along the s axis and from $h_t^0 \equiv h_t - \Delta h_t$ to $h_t^1 \equiv h_t + \Delta h_t$ along the t axis of the shading frame; the function φ is shorthand for

$$\varphi(x, y, \gamma, \delta) = \frac{x \arctan\left(\frac{\gamma y}{\sqrt{x^2 + \gamma^2} \delta}\right)}{\sqrt{x^2 + \gamma^2}}.$$