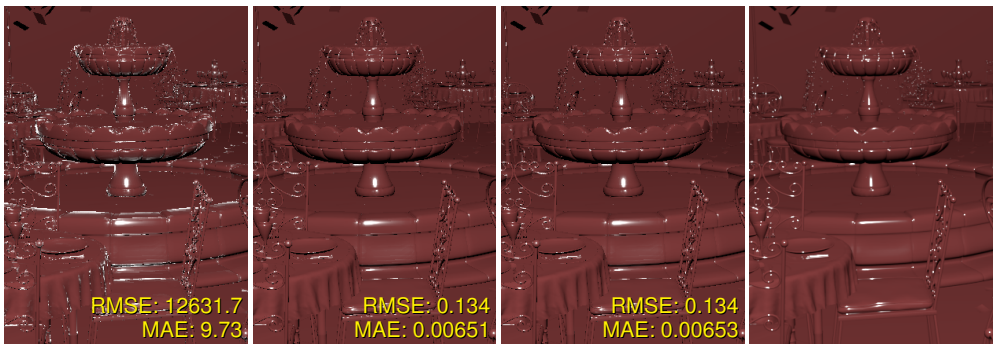
Stable Geometric Specular Antialiasing with Projected-Space NDF Filtering

Yusuke Tokuyoshi
Intel Corporation*

Anton S. Kaplanyan
Facebook Reality Labs



(a) Previous (b) Ours (c) Ours (approximation) (d) Reference

Figure 1. Geometric specular antialiasing using non-axis-aligned filtering for the GGX NDF with roughness $\alpha = 0.01$. Previous slope-space filtering (a) produces undesirable artifacts on rims, while our projected-space filtering (b) and its practical approximation (c) do not.

Abstract

Shading filtering proposed by Kaplanyan et al. is a simple solution for specular aliasing. It filters a distribution of microfacet normals in the domain of microfacet slopes by estimating the filtering kernel using derivatives of a halfway vector between incident and outgoing directions. However, for real-time rendering, this approach can produce noticeable artifacts because of an estimation error of derivatives. For forward rendering, this estimation error is increased significantly at grazing angles and near edges. The present work improves the quality of the original technique, while decreasing the complexity of the code at the same time. To reduce the error, we introduce a filtering method in the domain of orthographically projected microfacet normals and a practical approximation of this filtering method. In addition, we optimize the calculation of an isotropic filter kernel used for deferred rendering by applying the proposed projected-space filtering. As our implementation is simpler than the original

*now at AMD Japan Ltd.

 ISSN 2331-7418

method, it is easier to integrate in time-sensitive applications, such as game engines, while at the same time improving the filtering quality.

1. Introduction

Rendering images in the presence of both highly specular materials and complex geometry is a challenging task. A high-quality image requires finding all tiny speckles induced by geometric shapes as well as elongated anisotropic highlights that can often have areas much smaller than the footprint of a pixel. In addition, the presence of high-frequency shading surfaces, such as normal maps, bump maps, or displacement maps, amplifies this problem further. Moreover, as the roughness of the material is decreased, these highlights increase their intensity, while their area decreases. This makes them not only harder to find with pointwise samples used in rendering, but also less forgiving when missed due to their high contribution. These challenges make such effects very challenging to render in a temporally stable manner from one frame to another even with hundreds of samples spent per pixel.

In a real-time setting, rendering small specular highlights becomes even more challenging, because of a typical budget of only one shading sample per pixel. If, in addition, a pixel's area covers a larger solid angle, such as in VR head-mounted displays, the relative area of the highlight with respect to the pixel footprint gets even smaller, reducing the chances of finding the highlight with a single shading sample. Such high-intensity speckles can appear once in many frames, when one of the samples occasionally lands inside the highlight. Temporal accumulation and supersampling techniques, such as temporal antialiasing [Karis 2014], usually completely remove such outlier highlights, changing the material appearance.

In order to alleviate this problem, multiple prefiltering techniques have been introduced. They typically work by analyzing the local proximity around the shading point and filtering the highlight in the area with respect to the footprint covered by the shaded pixel, respecting variations of the surface and shading parameters. One such recent technique is geometric specular antialiasing that employs the filtering of a microfacet normals distribution function (NDF) [Kaplanyan et al. 2016]. This method is practically used in multiple game engines (e.g., Amazon Lumberyard [Chen 2017], Unity 2018.2+ [2018]) and shipped games. While providing high-quality filtering for smooth specular materials and varying geometry, the method also introduces instabilities and artifacts at grazing angles, especially when estimated using GPU quad derivatives. This work improves the quality of the original geometric specular antialiasing method (Figure 1), while simultaneously decreasing the complexity of the code. This paper builds upon our previous work [2019] and further introduces a novel filtering space for stable geometric specular antialiasing.

- This work analyzes the increasing derivatives-estimation error at grazing angles for geometric specular antialiasing in forward rendering (Section 3).

Symbol	Description
$\mathbf{i}$	Unit vector of incident direction
$\mathbf{o}$	Unit vector of outgoing direction
$\mathbf{n}$	Unit vector of shading normal at the surface point
$\mathbf{h}$	Unit halfway vector between $\mathbf{i}$ and $\mathbf{o}$
$[h_x, h_y, h_z]$	Unit halfway vector in tangent space
$D(\mathbf{h})$	Normals distribution function (NDF)
α	Isotropic roughness parameter in slope space ($\alpha \in [0, 1]$)
$\bar{\alpha}$	Filtered isotropic roughness parameter in slope space
α_x, α_y	Anisotropic roughness parameters in slope space ($[\alpha_x, \alpha_y] \in [0, 1]^2$)
$\mathbf{A}$	2×2 roughness matrix in slope space
$\bar{\mathbf{A}}$	Filtered 2×2 roughness matrix in slope space
$\mathbf{B}$	2×2 roughness matrix in the $[h_x, h_y]$ space
$\bar{\mathbf{B}}$	Filtered 2×2 roughness matrix in the $[h_x, h_y]$ space
$\mathbf{I}$	2×2 identity matrix
$\mathbf{\Sigma}_{\parallel}$	2×2 covariance matrix to represent the filter kernel in slope space
$\mathbf{\Sigma}_{\perp}$	2×2 covariance matrix to represent the filter kernel in the $[h_x, h_y]$ space
$\Delta\mathbf{h}_u^{\parallel}, \Delta\mathbf{h}_v^{\parallel}$	Derivatives of the halfvector in slope space
$\Delta\mathbf{h}_u^{\perp}, \Delta\mathbf{h}_v^{\perp}$	Derivatives of the halfvector in the $[h_x, h_y]$ space
$\lambda_{\min}, \lambda_{\max}$	Minimum and maximum eigenvalues of $\mathbf{\Sigma}_{\perp}$

Table 1. Notation used throughout the paper.

- To alleviate the above problem, we introduce an orthographically projected space for NDF filtering and a practical approximation of this filtering (Section 4).
- We also present a simple roughness calculation for deferred rendering based on the proposed space (Section 5).

We also demonstrate these improvements in a comparison with the previous work.

2. Background

2.1. Non-Axis-Aligned Anisotropic Microfacet BRDFs

The microfacet bidirectional reflectance distribution function (BRDF) model [Cook and Torrance 1982] is defined as

$$f(\mathbf{i}, \mathbf{o}) = \frac{F(\mathbf{i} \cdot \mathbf{h}) G_2(\mathbf{i}, \mathbf{o}) D(\mathbf{h})}{4|\mathbf{i} \cdot \mathbf{n}||\mathbf{o} \cdot \mathbf{n}|},$$

where $\mathbf{i}$ and $\mathbf{o}$ are incoming and outgoing directions, $\mathbf{h} = (\mathbf{i} + \mathbf{o})/\|\mathbf{i} + \mathbf{o}\|$ is the halfvector, $\mathbf{n}$ is the shading normal, $F(\mathbf{i} \cdot \mathbf{h})$ is the Fresnel factor, $G_2(\mathbf{i}, \mathbf{o})$ is the masking-shadowing function, and $D(\mathbf{h})$ is the NDF. See Table 1 for notation. Two

common NDFs used in renderers and game engines are the Beckmann distribution [1963] and the GGX distribution [Walter et al. 2007]. Although microfacet BRDFs are typically used with axis-aligned anisotropy, these BRDFs are straightforward to generalize for non-axis-aligned anisotropy [Heitz 2014]. The generalized Beckmann NDF is defined as

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}h_z^4}\exp\left(-\left[\frac{h_x}{h_z},\frac{h_y}{h_z}\right]\mathbf{A}^{-1}\left[\frac{h_x}{h_z},\frac{h_y}{h_z}\right]^\top\right),$$

where $[h_x, h_y, h_z]$ is the halfvector $\mathbf{h}$ in tangent space, and $\chi^+(h_z)$ is the Heaviside function: 1 if $h_z > 0$; otherwise 0. The matrix $\mathbf{A}$ is a 2×2 nonnegative-definite matrix to represent the roughness for non-axis-aligned anisotropy, which is two times the covariance matrix of the slope-space distribution. In this paper, we refer to this matrix as the *roughness matrix*. If surfaces are modeled with two roughness parameters α_x and α_y along the tangent and bitangent axes for the axis-aligned anisotropy, the roughness matrix is given as $\mathbf{A} = \begin{bmatrix}\alpha_x^2 & 0\\ 0 & \alpha_y^2\end{bmatrix}$. The GGX NDF is also written using $\mathbf{A}$ as a roughness parameter for non-axis-aligned anisotropy as follows:

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}\left([h_x,h_y]\,\mathbf{A}^{-1}\,[h_x,h_y]^\top + h_z^2\right)^2}. \tag{1}$$

For the Smith microsurface model [1967], the form of the masking-shadowing function $G_2(\mathbf{i},\mathbf{o})$ depends on the NDF model. For details on $G_2(\mathbf{i},\mathbf{o})$ for the non-axis-aligned anisotropic GGX NDF, please refer to Appendix A.

2.2. Slope-Space NDF Filtering

For geometric specular antialiasing, initially the Beckmann NDF is assumed and then it is filtered using an anisotropic Gaussian kernel in the slope domain. The filter kernel is given as a covariance matrix $\mathbf{\Sigma}_\parallel$ calculated for each pixel as follows:

$$\mathbf{\Sigma}_\parallel = \sigma^2\begin{bmatrix}\Delta\mathbf{h}_u^\parallel\\ \Delta\mathbf{h}_v^\parallel\end{bmatrix}^\top\begin{bmatrix}\Delta\mathbf{h}_u^\parallel\\ \Delta\mathbf{h}_v^\parallel\end{bmatrix},$$

where $\sigma^2 = \frac{1}{2\pi}$ is the variance of the pixel filter kernel in image space measured in pixels,* and $\Delta\mathbf{h}_u^\parallel$ and $\Delta\mathbf{h}_v^\parallel$ are the derivatives of the halfvector in slope space with respect to image-space axial pixel offsets. The filtering process is a convolution of the estimated kernel with the NDF of the material. Since the Beckmann NDF is a 2D Gaussian distribution in slope space, this filtering becomes a convolution of two Gaussian distributions, which has a simple closed-form solution. Hence, the resulting

*Previous work [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan 2019] used $\sigma^2 = 0.25$, but this paper uses $\sigma^2 = \frac{1}{2\pi}$. Please refer to Appendix B for details of our parameter setting.

filtered NDF is also an anisotropic Beckmann NDF that uses the following 2×2 matrix as its roughness parameter:

$$\bar{\mathbf{A}} = \mathbf{A} + 2\boldsymbol{\Sigma}_{\|}.$$

If the surfaces are modeled with axis-aligned anisotropy $\mathbf{A} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix}$, the filtered roughness matrix for this case is

$$\bar{\mathbf{A}} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix} + 2\boldsymbol{\Sigma}_{\|}. \tag{2}$$

It was also shown that this filtered roughness matrix can be used to approximate the filtering of the GGX NDF. Please see the original work [Kaplanyan et al. 2016] for more details.

3. Error Analysis for Derivative Estimation

Derivative Estimation. For real-time computer graphics APIs (e.g., DirectX® and OpenGL®), derivatives can be estimated in a pixel shader using intrinsic functions (e.g., `ddx` and `ddy` in HLSL). These intrinsics compute the difference between values of two adjacent pixels in a 2×2 shading quad. However, this rough estimation produces a numerical error (Figure 2(a)) especially for grazing angles of the view direction $\mathbf{o}$. To suppress artifacts caused by this error, Kaplanyan et al. [2016] used a biased axis-aligned rectangular filtering technique. They also clamped the bandwidth of their rectangular kernel. However, for the GGX NDF, artifacts can still be noticeable at grazing angles as shown in Figure 2(b).

(a) Non-axis-aligned filtering (b) Axis-aligned filtering (c) Reference

Figure 2. Previous slope-space NDF filtering methods for the SPONZA scene (closeups). (a) While NDF filtering is formulated using the non-axis-aligned filter kernel $\Sigma_{\|}$, it is numerically unstable in practice. (b) A biased axis-aligned filter kernel suppresses undesirable artifacts by blurring them. However, blurred artifacts can still be noticeable for the GGX NDF.

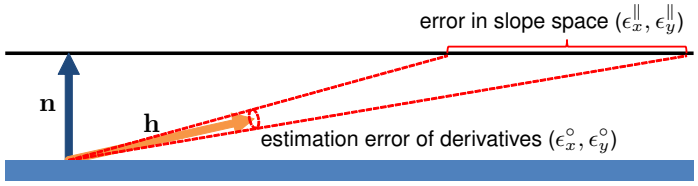


Figure 3. Estimation error of derivatives of the halfvector $\mathbf{h}$ is increased in slope space for grazing $\mathbf{h}$. This error can become larger than the magnitude of the slope of $\mathbf{h}$, and thus inappropriate filtering is performed.

Error Analysis. In this paper, we show that the above estimation error is significantly increased by projecting the halfvector $\mathbf{h}$ into slope space (Figure 3). This increase of the error is represented using the Jacobian matrix of the projection as follows:

$$\begin{bmatrix} \epsilon_x^{\parallel} \\ \epsilon_y^{\parallel} \end{bmatrix} = \mathbf{J}_{\circ\to\parallel} \begin{bmatrix} \epsilon_x^{\circ} \\ \epsilon_y^{\circ} \end{bmatrix},$$

where $\epsilon_x^{\parallel}$ and $\epsilon_y^{\parallel}$ are the errors in slope space: ϵ_x° is the error on the great circle passing through the halfvector $\mathbf{h}$ and normal $\mathbf{n}$; ϵ_y° is the error on the great circle passing through the halfvector $\mathbf{h}$ and $\frac{\mathbf{n}\times\mathbf{h}}{\|\mathbf{n}\times\mathbf{h}\|}$. The Jacobian matrix $\mathbf{J}_{\circ\to\parallel}$ of the transformation from spherical space to slope space is given by

$$\mathbf{J}_{\circ\to\parallel} = -\frac{1}{h_z^2\sqrt{1-h_z^2}} \begin{bmatrix} h_x & -h_y h_z \\ h_y & h_x h_z \end{bmatrix}.$$

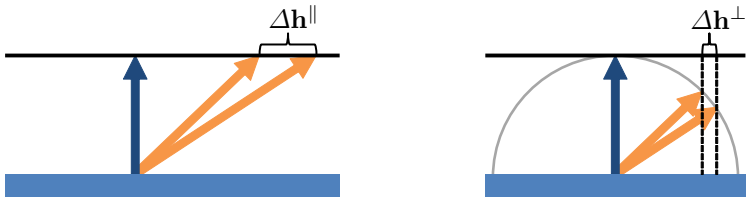
For derivation, please refer to Appendix C. The determinant of this Jacobian matrix is

$$\det\left(\mathbf{J}_{\circ\to\parallel}\right) = \frac{1}{h_z^3} \geq 1.$$

The magnitude of the error in slope space can be larger than the magnitude of the slope of the halfvector for small values of h_z. Hence, significant artifacts are induced for grazing halfvectors due to inappropriate filtering. These artifacts are noticeable especially for the GGX NDF, because the GGX distribution has a heavier tail than the Beckmann distribution. In addition, since this error can produce an inordinately elongated kernel, a noticeable precision error is produced later, when evaluating the filtered non-axis-aligned BRDF (please see Appendix A.4).

4. NDF Filtering in the Orthographically Projected Space

As was shown using Jacobian analysis, artifacts appear when the halfvector approaches grazing angles. However, NDF filtering of a smooth specular material is usually unnecessary for these grazing halfvectors because they do not produce highlights. For rough materials, filtering can be disabled all together, since it does not have an effect.



(a) Previous derivatives (b) Our derivatives

Figure 4. While the previous method (a) estimates the derivatives in slope space, our method (b) estimates the derivatives on the projected plane to reduce the filtering error.

Therefore, we introduce a projection of the filtering space to shrink the derivatives for grazing angles by assuming the GGX NDF. By shrinking the derivatives, we are also able to shrink the estimation error of the derivatives.

4.1. Proposed Projection

Instead of the projection into slope space for NDF filtering, we propose the orthographic projection onto the $[h_x, h_y]$ plane (Figure 4). Let $\Delta\mathbf{h}_u^{\perp}$ and $\Delta\mathbf{h}_v^{\perp}$ be the derivatives of $[h_x, h_y]$ (please see Kaplanyan et al. [2014] for details). Then our filter kernel is given by the following covariance matrix:

$$\boxed{\mathbf{\Sigma}_{\perp} = \sigma^2 \begin{bmatrix} \Delta\mathbf{h}_u^{\perp} \\ \Delta\mathbf{h}_v^{\perp} \end{bmatrix}^{\top} \begin{bmatrix} \Delta\mathbf{h}_u^{\perp} \\ \Delta\mathbf{h}_v^{\perp} \end{bmatrix}}. \tag{3}$$

This orthographic projection is simply implemented by removing a division code from the projection into slope space (please see Listings 2 and 3). The Jacobian matrix of this projection is given by

$$\mathbf{J}_{\circ\to\perp} = \frac{1}{\sqrt{1-h_z^2}} \begin{bmatrix} h_x h_z & -h_y \\ h_y h_z & h_x \end{bmatrix}.$$

The determinant of this Jacobian matrix is

$$\det\left(\mathbf{J}_{\circ\to\perp}\right) = h_z \leq 1.$$

Therefore, this projection avoids the increase of the error for grazing halfvectors.

4.2. Projected-Space Filtering for the GGX NDF

This section introduces an NDF filtering method in the orthographically projected space. Although this projected-space filtering is not practical (due to high cost and numerical limitations), we derive a practical approximation from this method in Section 4.3. To perform NDF filtering in the orthographically projected space, we transform the NDF from slope space to the projected space as shown in Figure 5. The GGX

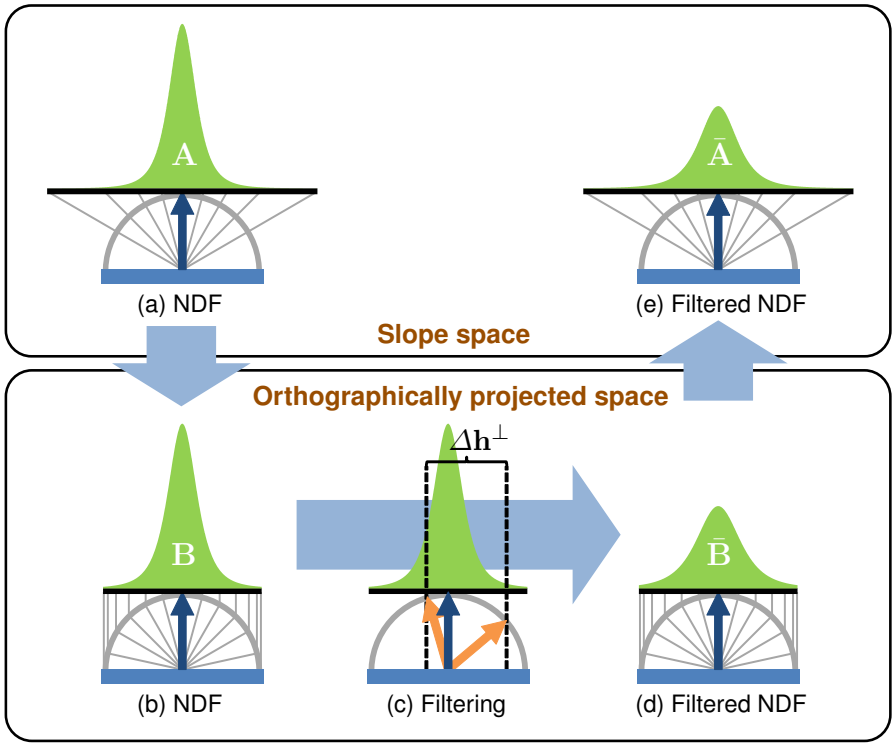


Figure 5. Our projected-space filtering for the GGX NDF. (a) The GGX NDF is a bell-shaped distribution P in slope space and parameterized with a roughness matrix $\mathbf{A}$. (b) Instead of this slope space, we map the NDF into a plane using orthographic projection. (c) In this projected space, the GGX NDF is an amplified P with a roughness matrix $\mathbf{B}$. Therefore, by approximating the distribution P and pixel footprint with Gaussians, we perform analytic NDF filtering in the projected space. Finally, the filtered NDF in the projected space (d) is mapped into slope space (e) to obtain the filtered roughness parameter $\bar{\mathbf{A}}$.

NDF (Equation (1)) is expressed as a distribution $P(\cdot)$ in slope space $\left[-\frac{h_x}{h_z}, -\frac{h_y}{h_z}\right]$ as follows:

$$D(\mathbf{h}) = P\left(-\frac{h_x}{h_z}, -\frac{h_y}{h_z}, \mathbf{A}\right) \chi^+(h_z) \det\left(\mathbf{J}_{\perp\to\parallel}\right),$$

where $\det\left(\mathbf{J}_{\perp\to\parallel}\right) = 1/h_z^4$ is the Jacobian of the transformation from the projected space $[h_x, h_y]$ to slope space $\left[-\frac{h_x}{h_z}, -\frac{h_y}{h_z}\right]$, and $P(\cdot)$ is a bivariate elliptical distribution given by

$$P(x, y, \mathbf{A}) = \frac{1}{\pi\sqrt{\det(\mathbf{A})}\left([x, y]\,\mathbf{A}^{-1}\,[x, y]^\top + 1\right)^2}.$$

For the GGX NDF, previous NDF filtering (Equation (2)) approximated the slope-space distribution $P\left(-\frac{h_x}{h_z}, -\frac{h_y}{h_z}, \mathbf{A}\right)$ with a Gaussian distribution. In this paper, we approximate the GGX NDF in the $[h_x, h_y]$ space with a Gaussian. Substituting $h_z^2 =$

$1-h_x^2-h_y^2$ into Equation (1), we equivalently rewrite the GGX NDF into the following form:

$$\begin{aligned} D(\mathbf{h}) &= \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}\left([h_x,h_y]\,\mathbf{A}^{-1}\,[h_x,h_y]^\top + 1 - h_x^2 - h_y^2\right)^2} \\ &= \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}\left([h_x,h_y]\,(\mathbf{A}^{-1}-\mathbf{I})\,[h_x,h_y]^\top + 1\right)^2} \\ &= \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}\left([h_x,h_y]\,\mathbf{B}^{-1}\,[h_x,h_y]^\top + 1\right)^2}, \end{aligned}$$

where matrix $\mathbf{B}$ is given by

$$\mathbf{B} = \left(\mathbf{A}^{-1} - \mathbf{I}\right)^{-1}. \tag{4}$$

Using this $\mathbf{B}$, we rewrite the GGX NDF into the following $[h_x, h_y]$-space distribution:

$$\begin{aligned} D(\mathbf{h}) &= \sqrt{\frac{\det(\mathbf{B})}{\det(\mathbf{A})}}\frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{B})}\left([h_x,h_y]\,\mathbf{B}^{-1}\,[h_x,h_y]^\top + 1\right)^2} \\ &= \boxed{\sqrt{\frac{\det(\mathbf{B})}{\det(\mathbf{A})}}P(h_x,h_y,\mathbf{B})\,\chi^+(h_z)}. \end{aligned} \tag{5}$$

In this form, $\mathbf{B}$ is the roughness matrix in the $[h_x, h_y]$ space, and $\sqrt{\frac{\det(\mathbf{B})}{\det(\mathbf{A})}}$ is the normalization factor for $[h_x, h_y]$ on a unit disk Ω as follows:

$$\sqrt{\frac{\det(\mathbf{B})}{\det(\mathbf{A})}} = \frac{1}{\iint_\Omega P(x,y,\mathbf{B})\mathrm{d}x\mathrm{d}y}.$$

As shown in Equation (5), the GGX NDF is represented using the distribution $P(\cdot)$ not only in slope space but also in the $[h_x, h_y]$ space. Therefore, we approximate $P(h_x, h_y, \mathbf{B})$ with a Gaussian distribution whose covariance matrix is $\frac{1}{2}\mathbf{B}$. Using the projected-space filter kernel whose covariance matrix is $\mathbf{\Sigma}_\perp$, NDF filtering is represented with the Gaussian convolution in the $[h_x, h_y]$ space. Although $[h_x, h_y]$ is a point on a unit disk Ω, we approximate the domain of the convolution with $\mathbb{R}^2$ to obtain the following analytical solution for the filtered roughness matrix:

$$\bar{\mathbf{B}} = \mathbf{B} + 2\mathbf{\Sigma}_\perp. \tag{6}$$

The filtered slope-space roughness matrix $\bar{\mathbf{A}}$ is obtained from this projected-space roughness matrix $\bar{\mathbf{B}}$ as follows:

$$\bar{\mathbf{A}} = \left(\bar{\mathbf{B}}^{-1} + \mathbf{I}\right)^{-1}. \tag{7}$$

```
float2x2 NonAxisAlignedNDFFiltering(float3 halfvectorTS, float2 roughness2) {
 // Compute the derivatives of the halfvector in the projected space.
 float2 halfvector2D = halfvectorTS.xy;
 float2 deltaU = ddx(halfvector2D);
 float2 deltaV = ddy(halfvector2D);

 // Compute 2 * covariance matrix for the filter kernel (Eq. (3)).
 float SIGMA2 = 0.15915494;
 float2x2 delta = {deltaU, deltaV};
 float2x2 kernelRoughnessMat = 2.0 * SIGMA2 * mul(transpose(delta), delta);

 // Convert the roughness from slope space to the projected space (Eq. (4)).
 float2 projRoughness2 = roughness2 / (1.0 - roughness2);
 float2x2 projRoughnessMat = {projRoughness2.x, 0.0, 0.0, projRoughness2.y};

 // NDF filtering in the projected space (Eq. (6)).
 float2x2 filteredProjRoughnessMat = projRoughnessMat + kernelRoughnessMat;

 // Convert the roughness from the projected space to slope space (Eq. (7)).
 // This implementation is optimized based on Appendix D.
 // For numerical stability, the determinant is clamped with the lower bound.
 float detMin = projRoughness2.x * projRoughness2.y;
 float det = max(determinant(filteredProjRoughnessMat), detMin);
 float2x2 m = filteredProjRoughnessMat / det + float2x2(1.0, 0.0, 0.0, 1.0);
 float2x2 filteredRoughnessMat = m / max(determinant(m), 1.0);
 return filteredRoughnessMat;
}
```

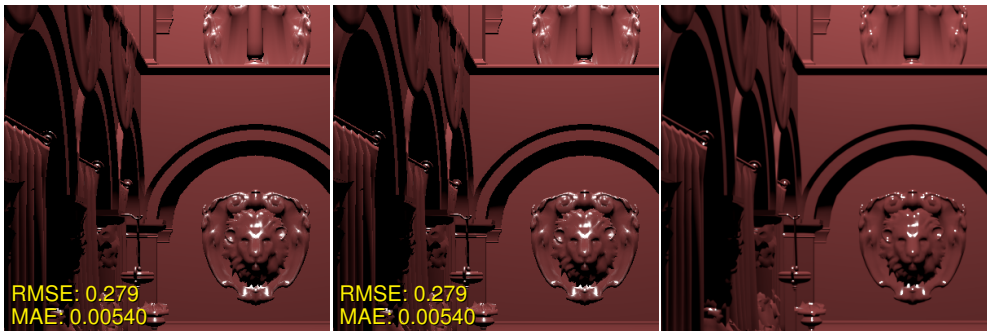
Listing 1. Our projected-space non-axis-aligned filtering (HLSL).

For the normalization factor of the filtered NDF, we use $\sqrt{\frac{\det(\bar{\mathbf{B}})}{\det(\bar{\mathbf{A}})}}$ instead of $\sqrt{\frac{\det(\mathbf{B})}{\det(\mathbf{A})}}$ to correct the approximation error caused by the change of the convolution domain. Hence, by mapping roughness parameters between slope space and projected space using Equations (4) and (7), we are able to perform projected-space filtering for the GGX NDF. This approach significantly reduces the filtering error from the previous slope-space method as shown in Figure 1. Listing 1 shows the HLSL code of our projected-space NDF filtering.

4.3. Practical Approximation

Although our projected-space filtering produces less error than slope-space filtering, the formulation of Section 4.2 is more expensive than slope-space filtering because of the roughness mapping (Equations (4) and (7)). In addition, our formulation assumes invertible matrices. Therefore, $\alpha_x = 1$ and $\alpha_y = 1$ are not supported. The assumption is also violated when both surface roughness and derivatives are zero. Even if the roughness is non-zero, our roughness mapping can induce an additional numerical error for extremely small roughness (e.g., $\alpha_x = 0.000001$ and $\alpha_y = 0.000001$). To

(a) Non-axis-aligned filtering



(b) Axis-aligned filtering

Figure 6. Our projected-space filtering and its practical approximation using the non-axis-aligned filter kernel (a) and axis-aligned filter kernel (b) for the SPONZA scene (closeups). The approximation error is negligible in our experiments.

reduce the computational burden and avoid those special cases, we introduce a practical approximation by assuming a small filter kernel $\mathbf{\Sigma}_\perp$. Substituting Equations (4) and (6) into Equation (7), the filtered slope-space roughness matrix is given by

$$\bar{\mathbf{A}} = \left(\left(\left(\mathbf{A}^{-1} - \mathbf{I} \right)^{-1} + 2\mathbf{\Sigma}_\perp \right)^{-1} + \mathbf{I} \right)^{-1}. \tag{8}$$

Assuming elements of $\mathbf{\Sigma}_\perp$ are small, we approximate Equation (8) by the following equation:

$$\bar{\mathbf{A}} \approx \mathbf{A} + 2\mathbf{\Sigma}_\perp.$$

Let $\mathbf{A} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix}$, then we have

$$\boxed{\bar{\mathbf{A}} \approx \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix} + 2\mathbf{\Sigma}_\perp}. \tag{9}$$

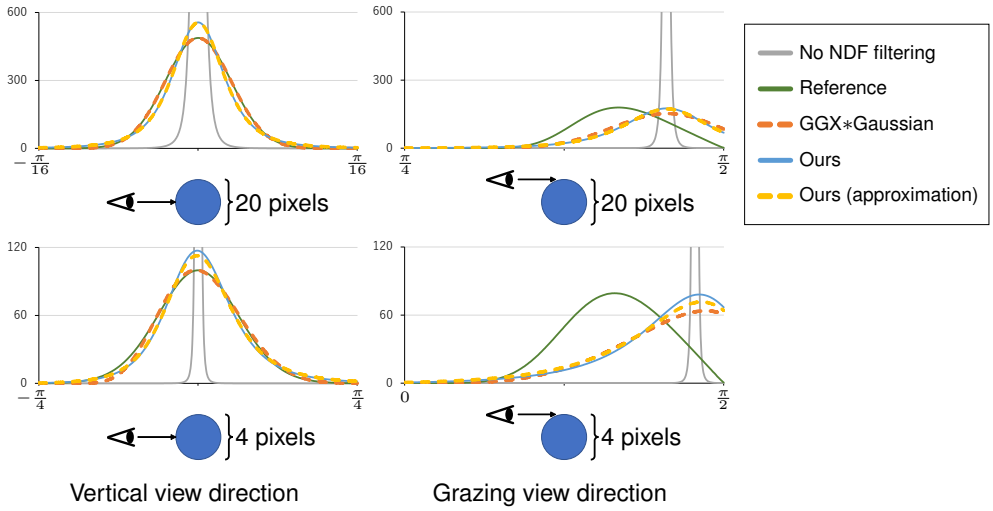


Figure 7. Plots of filtered GGX NDFs on a cylinder object. The horizontal axis is the angle between the view direction and halfvector. The dashed orange line is a numerical filtering that approximates only a pixel footprint with a Gaussian in the projected space. Our projected-space filtering (solid blue line) approximates both the NDF and pixel footprint to perform analytical filtering. We introduce a further approximation (yellow dashed line, Equation (9)), but the difference from the projected-space filtering is negligible.

Figure 6 and Figure 7 show the filtering results of our approximation and plots of filtered NDFs. This approximation can induce overfiltering when the filter kernel $\mathbf{\Sigma}_\perp$ is not small. However, the overfiltering error is smaller than previous slope-space filtering, since our approximation acts as slope-space filtering using a narrower filter kernel $\mathbf{\Sigma}_\perp$ than the previous kernel $\mathbf{\Sigma}_\parallel$ for shallower halfvector angles (Figure 8). This narrower kernel for shallower angles does not produce significant error because the tail

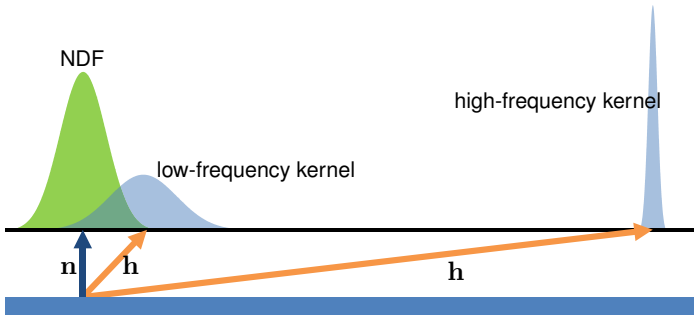


Figure 8. Our practical approximation acts as slope-space filtering with a higher-frequency filter kernel for a shallower halfvector angle. This filter kernel does not induce noticeable underfiltering error in practice, because a grazing halfvector angle is unlikely to produce specular highlights (i.e., noticeable aliasing).

```
float2x2 NonAxisAlignedNDFFiltering(float3 halfvectorTS, float2 roughness2) {
 // Compute the derivatives of the halfvector in the projected space.
 float2 halfvector2D = halfvectorTS.xy / abs(halfvectorTS.z);
 float2 deltaU = ddx(halfvector2D);
 float2 deltaV = ddy(halfvector2D);

 // Compute 2 * covariance matrix for the filter kernel (Eq. (3)).
 float SIGMA2 = 0.15915494;
 float2x2 delta = {deltaU, deltaV};
 float2x2 kernelRoughnessMat = 2.0 * SIGMA2 * mul(transpose(delta), delta);

 // Approximate NDF filtering (Eq. (9)).
 float2x2 roughnessMat = {roughness2.x, 0.0, 0.0, roughness2.y};
 float2x2 filteredRoughnessMat = roughnessMat + kernelRoughnessMat;
 return filteredRoughnessMat;
}
```

Listing 2. Practical approximation of our projected-space non-axis-aligned filtering (HLSL). This implementation is the same as Kaplanyan et al.'s slope-space filtering, except the red code is removed.

of the NDF is low frequency in the pixel footprint. On the other hand, our approximation significantly reduces undesirable artifacts without increasing the complexity of the shader code. This is because computation of $\mathbf{\Sigma}_\perp$ is simpler than slope-space filtering as shown in Listing 2.

Clamping the kernel size. In a similar manner to previous Kaplanyan [2016]'s implementation, we can clamp the kernel size in order to reduce the potential overfilter-

```
float2 AxisAlignedNDFFiltering(float3 halfvectorTS, float2 roughness2) {
 // Compute the bounding rectangle of halfvector derivatives.
 float2 halfvector2D = halfvectorTS.xy / abs(halfvectorTS.z);
 float2 bounds = fwidth(halfvector2D);

 // Compute an axis-aligned filter kernel from the bounding rectangle.
 float SIGMA2 = 0.15915494;
 float2 kernelRoughness2 = 2.0 * SIGMA2 * (bounds * bounds);

 // Approximate NDF filtering (Eq. (9)).
 // We clamp the kernel size to avoid overfiltering.
 float KAPPA = 0.18;
 float2 clampedKernelRoughness2 = min(kernelRoughness2, KAPPA);
 float2 filteredRoughness2 = saturate(roughness2 + clampedKernelRoughness2);
 return filteredRoughness2;
}
```

Listing 3. Practical approximation of our projected-space axis-aligned filtering (HLSL). This implementation (with clamping) is the same as Kaplanyan et al.'s slope-space filtering, except the red code is removed.

ing error of our approximation for large kernels. However, it can be computationally expensive for non-axis-aligned filtering, because it has to clamp the eigenvalues of $\mathbf{\Sigma}_\perp$ using eigendecomposition. On the other hand, it is inexpensive for biased axis-aligned filtering, since the eigenvalue is equal to the variance of the kernel for each axis. The HLSL code of our axis-aligned filtering with this clamping approach is shown in Listing 3.

5. Optimization for Deferred Rendering

5.1. NDF Filtering for Deferred Rendering

For deferred rendering, the use of halfvectors is not practical, because the inexpensive derivative estimation is usable only in the pixel shader. In addition, light sources are unknown for the G-buffer rendering pass. We also employ the approximation used for deferred rendering from Kaplanyan et al. [2016]. This approximation uses an average normal $\hat{\mathbf{n}}$ within the shading quad instead of the halfvector $\mathbf{h}$ for derivative estimation by assuming the worst case for a distant light source and distant eye position as follows:

$$\mathbf{\Sigma}_\perp = \sigma^2 \begin{bmatrix} \Delta\hat{\mathbf{n}}_u^\perp \\ \Delta\hat{\mathbf{n}}_v^\perp \end{bmatrix}^\top \begin{bmatrix} \Delta\hat{\mathbf{n}}_u^\perp \\ \Delta\hat{\mathbf{n}}_v^\perp \end{bmatrix},$$

where $\Delta\hat{\mathbf{n}}_u^\perp$ and $\Delta\hat{\mathbf{n}}_v^\perp$ are derivatives on the average normal $\hat{\mathbf{n}}$ on the projected space. For a compact G-buffer, since a single scalar roughness parameter is often required, Kaplanyan et al. [2016] proposed to use the maximum roughness of their rectangular filter kernel. In this section, we discuss alternative options for the case of a single scalar roughness. Using the projected-space derivatives from Section 4, we explore simple isotropic filter kernels.

The computation cost of NDF filtering is typically not a bottleneck when rendering a G-buffer that is dominated by a large memory transfer cost. However, the NDF filtering technique for deferred rendering can also be desirable to use for forward rendering in practical game engines [Unity Technologies 2018]. The reason is that the computation cost of normal-based filtering is independent of the number of light sources, and it supports any real-time approximation techniques for various types of light sources (e.g., area lights and environment maps) as well as indirect illumination. Therefore, simplification of NDF filtering is not only beneficial to reduce the implementation cost, but also to improve the performance for in these practical applications.

5.2. Constraint for Conservative Isotropic Filtering

In order to completely remove specular aliasing, the bandwidth of the isotropic kernel must be conservatively wider than that of the tight anisotropic kernel (represented

by the covariance matrix $\mathbf{\Sigma}_\perp$). Kaplanyan et al. [2016] used the maximum roughness of a circumscribed rectangle to satisfy this constraint. On the other hand, this conservative kernel bandwidth should be as tight as possible to reduce overfiltering. Within this constraint and objective, the optimal kernel bandwidth can be obtained as the maximum eigenvalue of the covariance matrix $\mathbf{\Sigma}_\perp$. Let $\lambda_{\max}$ be the maximum eigenvalue. In this case, the roughness of the filtered isotropic NDF is given by

$$\bar{\alpha}^2 = \alpha^2 + \min\left(2\lambda_{\max}, \kappa\right), \tag{10}$$

where α is the original roughness parameter for the isotropic NDF, and $\kappa = 0.18$ is the clamping threshold used in the Kaplanyan et al. [2016]'s axis-aligned filtering to suppress the estimation error of derivatives. Although this eigenvalue $\lambda_{\max}$ can be calculated analytically, it is more expensive than the rectangular kernel-based approach. Therefore, we propose another option that satisfies the above constraint.

5.3. Simple Conservative Filtering

Instead of the maximum eigenvalue $\lambda_{\max}$, we can employ the sum of the minimum and maximum eigenvalues $\lambda_{\min} + \lambda_{\max}$ for conservative isotropic filtering. This sum is inexpensively obtained by the trace of $\mathbf{\Sigma}_\perp$ as follows:

$$\lambda_{\max} \leq \lambda_{\min} + \lambda_{\max} = \operatorname{tr}\left(\mathbf{\Sigma}_\perp\right) = \sigma^2 \left(\left\| \Delta \hat{\mathbf{n}}_u^\perp \right\|^2 + \left\| \Delta \hat{\mathbf{n}}_v^\perp \right\|^2 \right). \tag{11}$$

5.4. Optimization for Norms of Derivatives

Kaplanyan et al. [2016] computed the average normal $\hat{\mathbf{n}}$ using the average within the shading quad; in this paper we employ the average of two contiguous pixels for each screen axis as follows:

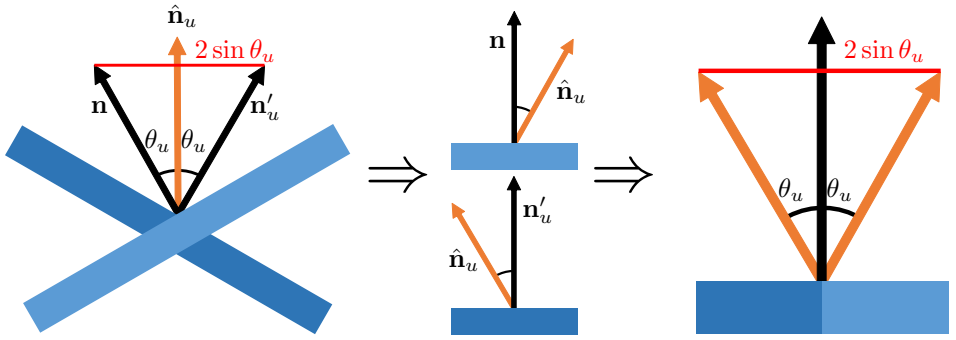
$$\hat{\mathbf{n}}_u = \frac{\mathbf{n} + \mathbf{n}'_u}{\|\mathbf{n} + \mathbf{n}'_u\|}, \qquad \hat{\mathbf{n}}_v = \frac{\mathbf{n} + \mathbf{n}'_v}{\|\mathbf{n} + \mathbf{n}'_v\|},$$

where $\mathbf{n}'_u$ and $\mathbf{n}'_v$ are normals at the neighboring pixels for each screen axis. For this case, $\hat{\mathbf{n}}_u$ is on the great circle defined by $\mathbf{n}$ and $\mathbf{n}'_u$, and the angle between $\mathbf{n}$ and $\hat{\mathbf{n}}_u$ is equal to the angle between $\mathbf{n}'_u$ and $\hat{\mathbf{n}}_u$. As shown in Figure 9, let this angle be θ_u, then the norm of the derivative of $\hat{\mathbf{n}}_u^\perp$ is given by

$$\left\| \Delta \hat{\mathbf{n}}_u^\perp \right\| = 2 \sin \theta_u = \|\mathbf{n} - \mathbf{n}'_u\|.$$

The same relation is yielded for $\mathbf{n}$, $\mathbf{n}'_v$, and $\hat{\mathbf{n}}_v$. Since $\mathbf{n} - \mathbf{n}'_u$ and $\mathbf{n} - \mathbf{n}'_v$ are equivalent to the derivatives of world-space normals $\Delta\mathbf{n}_u$ and $\Delta\mathbf{n}_v$, we derive

$$\left\| \Delta \hat{\mathbf{n}}_u^\perp \right\| = \|\Delta \mathbf{n}_u\|, \qquad \left\| \Delta \hat{\mathbf{n}}_v^\perp \right\| = \|\Delta \mathbf{n}_v\|. \tag{12}$$



(a) Average of two normals (b) Tangent space (c) Derivative in tangent space

Figure 9. If the average normal $\hat{\mathbf{n}}_u$ is computed using two normals $\mathbf{n}$ and $\mathbf{n}'_u$ of contiguous pixels (a), the length of the estimated derivative of the average normals in tangent space (c) is equal to the distance between $\mathbf{n}$ and $\mathbf{n}'_u$ (i.e., length of the derivative of world-space normals).

5.5. Optimized Conservative Filtering

By substituting Equation (12) in Equation (11), we obtain the following equation:

$$\lambda_{\min} + \lambda_{\max} = \sigma^2 \left(\|\Delta \mathbf{n}_u\|^2 + \|\Delta \mathbf{n}_v\|^2 \right).$$

Hence, our roughness for the filtered isotropic NDF is yielded as

$$\boxed{\bar{\alpha}^2 = \alpha^2 + \min\left(2\sigma^2 \left(\|\Delta \mathbf{n}_u\|^2 + \|\Delta \mathbf{n}_v\|^2 \right), \kappa\right)}. \tag{13}$$

Since this calculation uses world-space normals, the computation of the average normal and transformation into tangent space are unnecessary. Our implementation (Listing 5) is simpler than computing the rectangular kernel-based maximum roughness (Listing 4). In addition, since this roughness calculation is independent of tangent vectors, it performs robustly even if objects do not have valid tangent vectors. The visualization of the filtered roughness parameter is shown in Figure 10. For deferred rendering, there are no large differences between our method and Kaplanyan et al. [2016], while the proposed implementation is simpler.

5.6. Less Conservative Filtering

While conservative isotropic filtering removes specular aliasing, it induces overfiltering instead. Therefore, a smaller kernel than the conservative filtering might be more practical, though underfiltering can occur. We found the arithmetic mean of

```
float IsotropicNDFFiltering(float3 normal, float roughness2,
                            float3x3 tangentFrame, float2 pixelPosition) {
 float SIGMA2 = 0.15915494;
 float KAPPA = 0.18;
 float2 neighboringDir = 0.5 - 2.0 * frac(pixelPosition * 0.5);
 float3 deltaNormalX = ddx_fine(normal) * neighboringDir.x;
 float3 deltaNormalY = ddy_fine(normal) * neighboringDir.y;
 float3 avgNormal = normal + deltaNormalX + deltaNormalY;
 float3 avgNormalTS = mul(tangentFrame, avgNormal);
 float2 avgNormal2D = avgNormalTS.xy / abs(avgNormalTS.z);
 float2 bounds = fwidth(avgNormal2D);
 float maxWidth = max(bounds.x, bounds.y);
 float kernelRoughness2 = 2.0 * SIGMA2 * (maxWidth * maxWidth);
 float clampedKernelRoughness2 = min(kernelRoughness2, KAPPA);
 float filteredRoughness2 = saturate(roughness2 + clampedKernelRoughness2);
 return filteredRoughness2;
}
```

Listing 4. Previous conservative isotropic NDF filtering using the maximum width of the rectangular kernel for deferred rendering (HLSL).

```
float IsotropicNDFFiltering(float3 normal, float roughness2) {
 float SIGMA2 = 0.15915494;
 float KAPPA = 0.18;
 float3 dndu = ddx(normal);
 float3 dndv = ddy(normal);
 float kernelRoughness2 = 2.0 * SIGMA2 * (dot(dndu, dndu) + dot(dndv, dndv));
 float clampedKernelRoughness2 = min(kernelRoughness2, KAPPA);
 float filteredRoughness2 = saturate(roughness2 + clampedKernelRoughness2);
 return filteredRoughness2;
}
```

Listing 5. Our conservative isotropic NDF filtering for deferred rendering (HLSL). The red code is removed for our less conservative filtering.

(a) Kaplanyan et al. [2016] **(b)** Ours (Equation (13)) Difference: (a)−(b)

Figure 10. Visualization of roughness parameter $\bar{\alpha}$ of the filtered isotropic NDF for deferred rendering.

eigenvalues can be used to balance underfiltering and overfiltering as follows:

$$\begin{aligned}\bar{\alpha}^2 &= \alpha^2 + \min\left(2\left(\frac{\lambda_{\min}+\lambda_{\max}}{2}\right), \kappa\right) \\ &= \boxed{\alpha^2 + \min\left(\sigma^2\left(\|\Delta\mathbf{n}_u\|^2 + \|\Delta\mathbf{n}_v\|^2\right), \kappa\right)}.\end{aligned} \tag{14}$$

6. Results

Here we present the results of NDF filtering for the GGX microfacet BRDF. Similar results can be obtained for the Beckmann NDF, but we decided to focus on the GGX NDF as the most challenging case. All images are rendered at 1920×1080 pixels on an NVIDIA® GeForce® RTX 2080 GPU. The image quality is evaluated with the root-mean-squared error (RMSE) metric and mean absolute error (MAE) metric. In our experiments, reference images shown in Figure 11 are rendered using 16384 samples per pixel (spp). For these reference images, a Gaussian distribution with $\sigma^2 = \frac{1}{2\pi}$ is used for the pixel filter kernel. Rendering results using 1 spp without NDF filtering are shown in Figure 12.

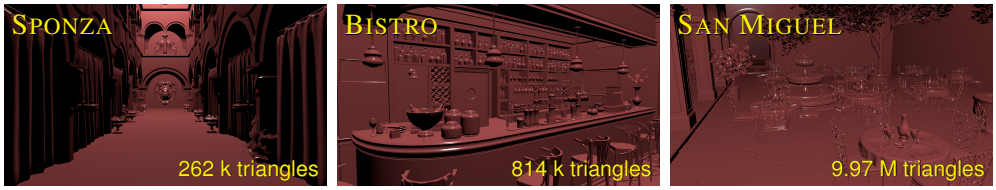


Figure 11. Reference images in our experiments (16384 spp).

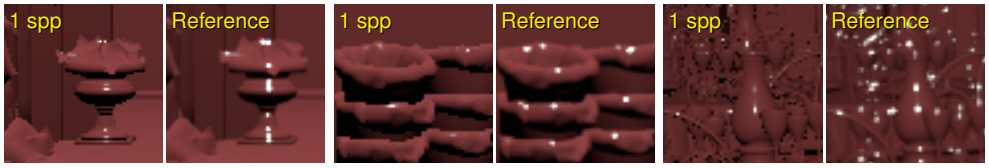


Figure 12. Closeups of rendered images without NDF filtering (1 spp).

Forward Rendering. Figures 13 and 14 demonstrate the improvements from the slope-space approach for non-axis-aligned filtering and axis-aligned filtering, respectively. For non-axis-aligned filtering, our projected-space filtering (Listing 1) reduces both the RMSE and MAE significantly. Our non-axis-aligned filtering is higher quality than axis-aligned filtering in terms of the RMSE and MAE metrics, while some pixels can still flicker in animation (please see the supplemental video). For such

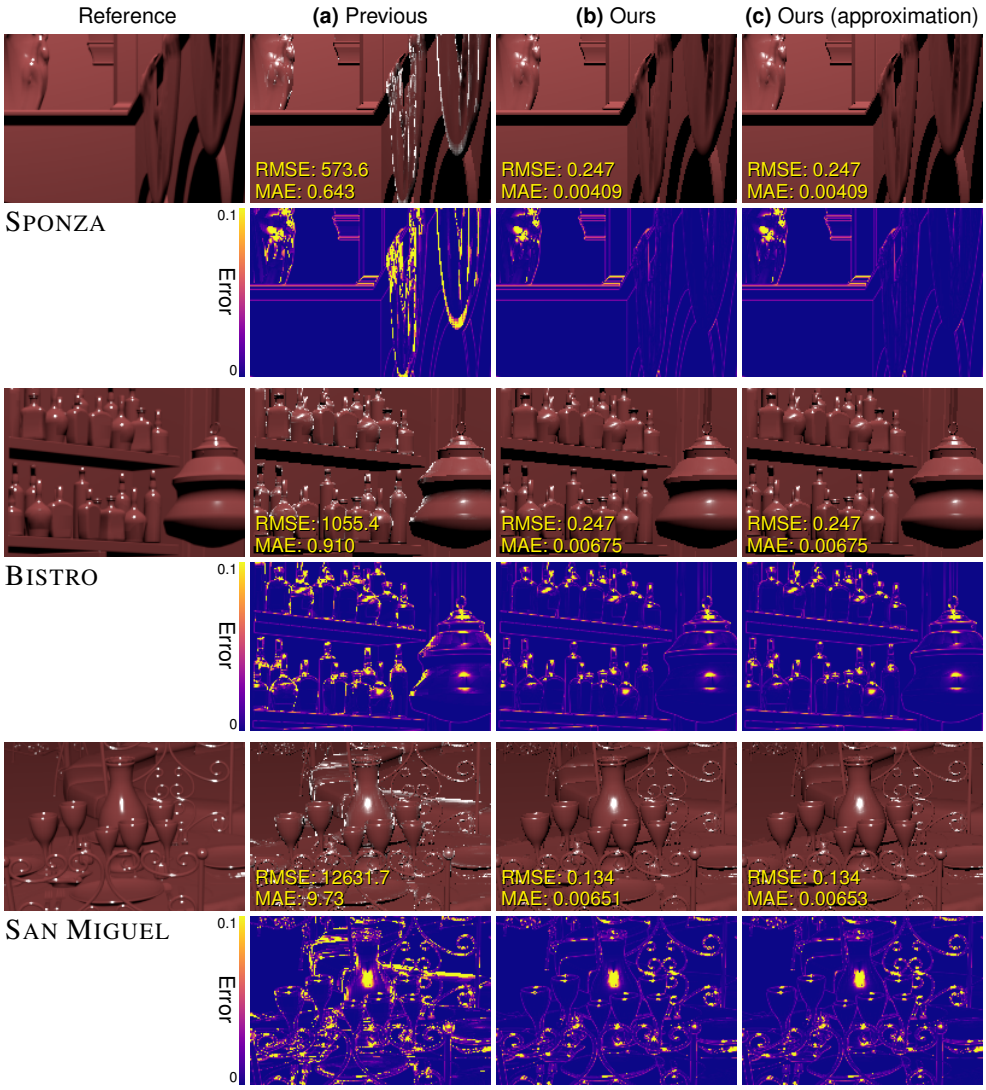


Figure 13. Quality comparison of non-axis-aligned filtering for forward rendering. Images are closeups of rendering results and the error from the reference. Compared to previous slope-space filtering (a), our projected-space filtering (b) and its approximation (c) reduce errors at grazing angles significantly.

dynamic scenes, biased axis-aligned filtering is more practical because of the temporal stability. Even for this axis-aligned filtering, our method avoids undesirable artifacts on grazing angles and reduces the MAE. Our practical approximation (Listings 2 and 3) produces almost the same results as our projected-space filtering, while the implementation is simpler than both slope-space and projected-space approaches.

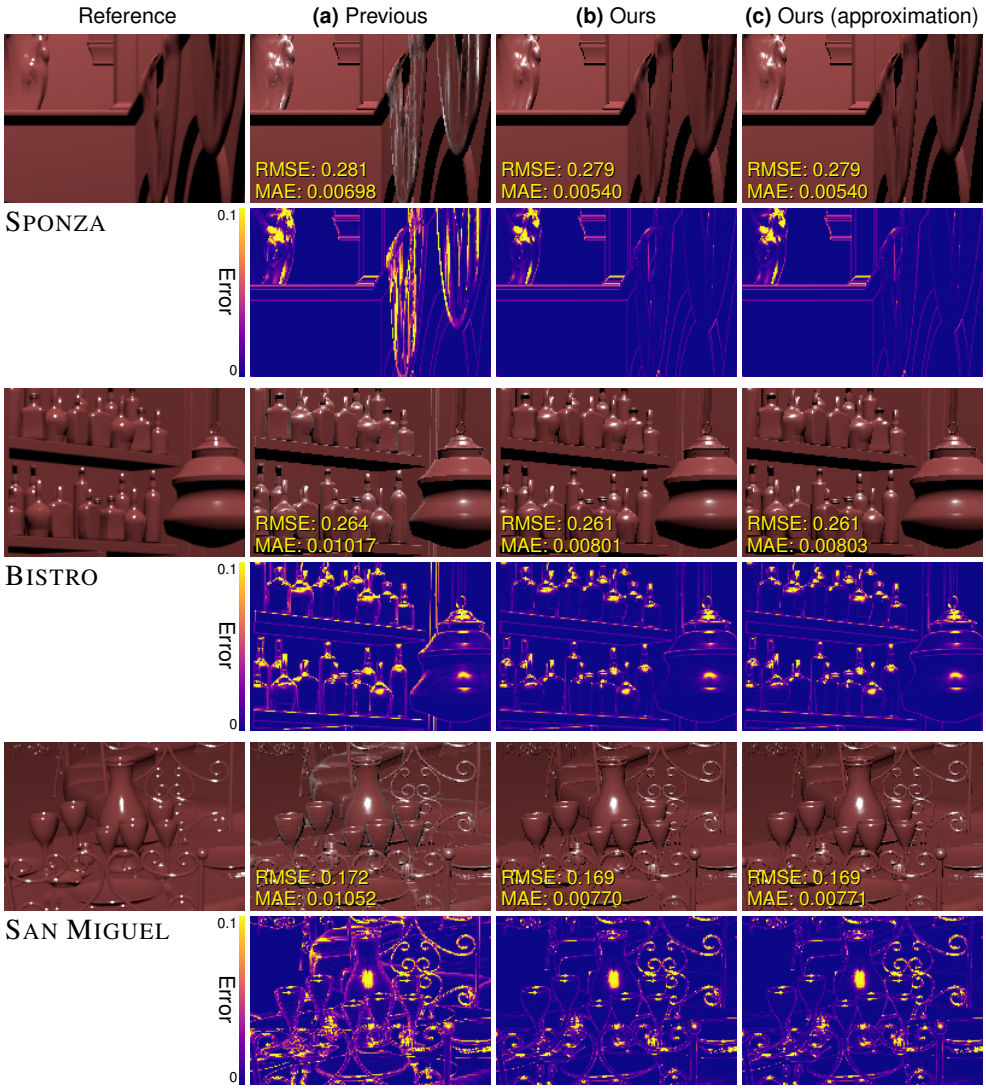


Figure 14. Quality comparison of axis-aligned filtering for forward rendering. Images are closeups of rendering results and the error from the reference. Our projected-space filtering (b) and its approximation (c) produce smaller MAE than previous slope-space filtering (a).

Deferred Rendering. In addition to the improvements of forward rendering, we propose a simplification of isotropic NDF filtering for deferred rendering. Our algorithm is shown in Listing 5, while the previous algorithm is provided in Listing 4. The quality comparison of these methods is shown in Figure 15. NDF filtering using the maximum eigenvalue (Equation (10)) has the smallest error in the constraint of conservative filtering in theory. The proposed conservative filtering (Equation (13)) produces slightly smaller error than the previous method, while our implementation is much simpler than the previous method and the above optimal approach. These

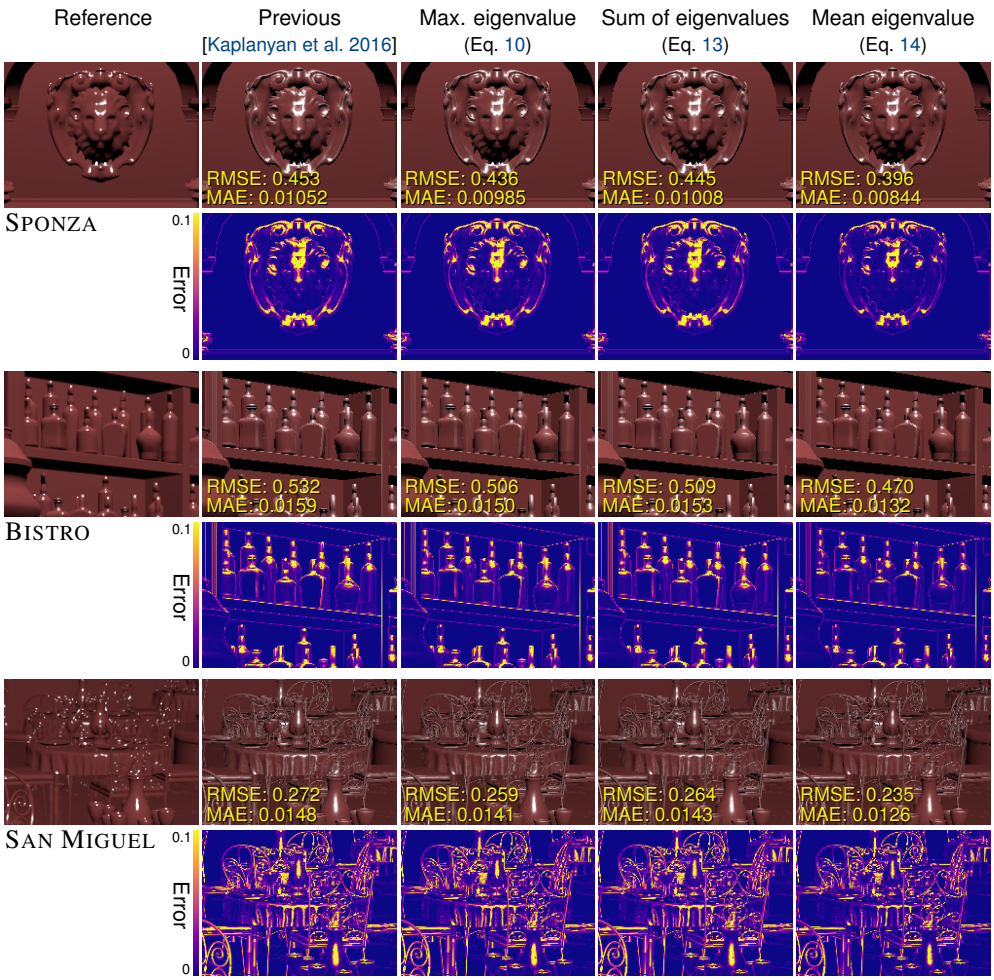


Figure 15. Quality comparison of isotropic NDF filtering for deferred rendering (closeups). Our less conservative filtering using the mean of eigenvalues (rightmost) produces the lowest RMSE and MAE in this experiment, while its implementation is the simplest.

techniques are conservative to avoid underfiltering, but they induce overfiltering instead. Our less conservative filtering (Equation (14)) reduces the error by balancing overfiltering and underfiltering. While less conservative filtering can produce aliasing artifacts slightly more than conservative filtering, it alleviates the change of material appearance caused by overfiltering (please see the supplemental video).

Performance. Table 2 shows the computational time for forward shading at 8K resolution (7680×4320 pixels) on the NVIDIA® GeForce® RTX 2080 GPU. Our method is not only applicable to high-end GPUs but also low power consumption platforms such as mobile devices. For such mobile applications, we also show the forward shading time for low-polygon scenes at 1920×1080 pixels on an Intel® Iris® Plus graph-

		SPONZA (261 k tris)	BISTRO (814 k tris)	SAN MIGUEL (9.97 M tris)
No NDF filtering		0.93	1.07	2.95
Non-axis-aligned filtering	Previous	1.45	1.54	3.42
	Ours	1.51	1.60	3.47
	Ours (approx)	1.44	1.54	3.41
Axis-aligned filtering	Previous	1.38	1.48	3.36
	Ours	1.38	1.49	3.36
	Ours (approx)	1.36	1.46	3.33
Normal-based isotropic filtering	Previous	1.47	1.57	3.44
	Max (Eq. (10))	1.54	1.63	3.50
	Sum (Eq. (13))	1.05	1.18	3.04
	Mean (Eq. (14))	1.05	1.18	3.04

Table 2. Forward shading time (ms) at 8K resolution (NVIDIA® GeForce® RTX 2080 GPU).

ics (integrated in an Intel® Core™ i5-1035G7 processor) in Table 3. As described in Section 5.1, since normal-based isotropic filtering for deferred rendering can also be used for forward rendering, this paper also evaluates the normal-based filtering for forward rendering. For non-axis-aligned and axis-aligned filtering, although the computation time of our projected-space filtering is slightly larger than the previous slope-space filtering, the approximated version of our method is almost the same as or slightly faster than the previous method. This is because our derivative estimation is simpler than the previous method. For normal-based isotropic NDF filtering, conservative filtering using the maximum eigenvalue is more expensive than the previous rectangular kernel-based approach. On the other hand, our simple filtering techniques using the sum of eigenvalues and mean of eigenvalues are significantly faster than the previous method, while they produce less error. The performance improvement of

		SPONZA (261 k tris)	BISTRO (814 k tris)
No NDF filtering		1.29	3.77
Non-axis-aligned filtering	Previous	1.79	4.28
	Ours	1.86	4.33
	Ours (approx)	1.78	4.24
Axis-aligned filtering	Previous	1.76	4.23
	Ours	1.78	4.26
	Ours (approx)	1.73	4.21
Normal-based isotropic filtering	Previous	1.87	4.35
	Max (Eq. (10))	1.92	4.39
	Sum (Eq. (13))	1.37	3.84
	Mean (Eq. (14))	1.37	3.84

Table 3. Forward shading time (ms) at 1920×1080 pixels (Intel® Iris® Plus graphics).

our simplification is effective when shaders are ALU bound. In this experiment, the computational overhead of our method is about 0.5 ms for non-axis-aligned filtering, about 0.4 ms for axis-aligned filtering, and about 0.1 ms for normal-based isotropic filtering.

7. Limitations

Our method is built upon Kaplanyan et al. [2016]'s work and inherits the usual limitations of this previous work. NDF filtering addresses only the aliasing caused by specular highlights, and aliasing caused by geometric discontinuities cannot be handled by this method. Real-time approximation of the pixel footprint introduces bias. In addition, the filtering of the GGX NDF is approximated by assuming a Gaussian distribution in the filtering space, therefore the GGX highlights can be overblurred due to this approximation. The method requires high-quality tangent frames. However, for our isotropic filtering, this limitation is reduced to high-quality shading normals. Unlike the previous method, our method is derived by assuming the GGX NDF. Therefore, the proposed filtering can have approximation error for the Beckmann NDF at grazing halfvectors. However, it is usually not a problem because aliasing is small for grazing halfvectors.

8. Conclusions

In this paper we have presented an error-reduction technique for NDF filtering. The rough derivative estimation produces a significant numerical error, since the error is increased due to the projection into slope space. To suppress this increase of the error, this paper introduces an orthographically projected space for NDF filtering whose filter kernel is narrower for a shallower halfvector angle. A practical approximation of this projected-space NDF filtering was also presented. Our approximation is simply implemented by estimating derivatives of a projected halfvector instead of a slope-space halfvector. In addition, we presented optimized isotropic NDF-filtering techniques for deferred and forward rendering based on this derivative estimation. Our method reduces the error as well as simplifies the shader code for geometric specular antialiasing.

Acknowledgements

We would like to thank F. Meinl and E. Bischoff for the SPONZA scene, the Amazon Lumberyard team for the BISTRO scene, and Guillermo M. Leal Llaguno for the SAN MIGUEL scene. These scenes are distributed by M. McGuire. We would also like to thank Intel's Advanced Rendering Technology team and the anonymous reviewers for their valuable comments and constructive suggestions.

A. Microsurface Model with Non-axis-aligned Anisotropy

A.1. Masking-shadowing Function

The Smith masking function [1967] is defined as $G_1(\mathbf{i}, \mathbf{h}) = \frac{\chi^+(\mathbf{i}\cdot\mathbf{h})}{1+\Lambda(\mathbf{i})}$. $\Lambda(\mathbf{i})$ is a function which depends on the NDF model. The height-correlated masking-shadowing function [Heitz 2014] is given as

$$G_2(\mathbf{i}, \mathbf{o}) = \frac{\chi^+(\mathbf{i}\cdot\mathbf{h})\,\chi^+(\mathbf{o}\cdot\mathbf{h})}{1+\Lambda(\mathbf{i})+\Lambda(\mathbf{o})}.$$

In this paper, $\Lambda(\mathbf{o})$ for the anisotropic GGX NDF model is described in the later subsections.

A.2. Axis-Aligned Anisotropic GGX Model

The axis-aligned anisotropic GGX NDF is defined as follows:

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\alpha_x\alpha_y\left(\frac{h_x^2}{\alpha_x^2}+\frac{h_y^2}{\alpha_y^2}+h_z^2\right)^2}.$$

For this NDF, the masking-shadowing function is obtained using the following function:

$$\Lambda(\mathbf{o}) = -0.5 + \frac{\sqrt{\alpha_x^2 o_x^2 + \alpha_y^2 o_y^2 + o_z^2}}{2|o_z|},$$

where $[o_x, o_y, o_z]$ is the outgoing direction $\mathbf{o}$ in tangent space.

A.3. Non-axis-aligned Anisotropic GGX Model

For specular antialiasing, we use the 2×2 roughness matrix $\mathbf{A}$ instead of α_x and α_y. The anisotropic NDFs can be generalized using this matrix as described in Section 2.1. For the non-axis-aligned anisotropic GGX NDF, the masking-shadowing function is obtained using the following function:

$$\Lambda(\mathbf{o}) = -0.5 + \frac{\sqrt{[o_x, o_y]\mathbf{A}[o_x, o_y]^\top + o_z^2}}{2|o_z|}.$$

For this microsurface model, the slope of a microsurface is stretched in the directions of the eigenvectors of the roughness matrix $\mathbf{A}$. The stretching scale for each eigenvector is the reciprocal square root of the eigenvalue of $\mathbf{A}$.

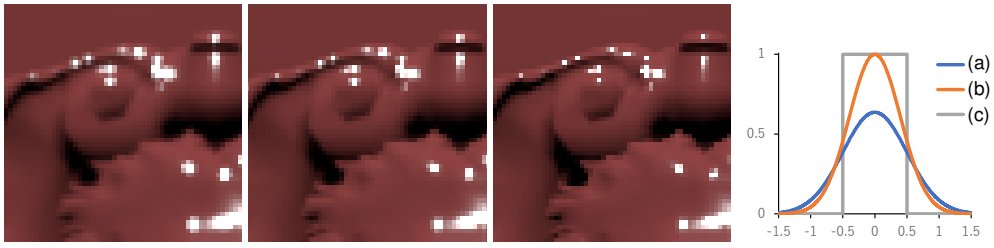
A.4. Practical Implementation for the NDF

The determinant $\det(\mathbf{A})$ can produce a large precision error due to floating-point arithmetic, especially when using an elongated kernel for NDF filtering. To improve the numerical stability, our method clamps $\det(\mathbf{A})$ to a small value τ for the NDF:

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\sqrt{\max(\det(\mathbf{A}), \tau)}\left([h_x, h_y]\mathbf{A}^{-1}[h_x, h_y]^\top + h_z^2\right)^2}.$$

To compute $\mathbf{A}^{-1}$, we also use this clamped determinant as follows:

$$\mathbf{A}^{-1} = \frac{\mathrm{adj}(\mathbf{A})}{\max(\det(\mathbf{A}), \tau)}.$$



(a) Gaussian filter ($\sigma^2 = 0.25$) (b) Gaussian filter ($\sigma^2 = \frac{1}{2\pi}$) (c) Box filter

Figure 16. Supersample antialiasing with different pixel filter kernels. The right most is the plots of the kernels. Compared to the Gaussian filter with $\sigma^2 = 0.25$ (a), the Gaussian filter with $\sigma^2 = \frac{1}{2\pi}$ (b) has less overblurring and is visually closer to the box filter (c).

For NDF filtering, since $\det(\bar{\mathbf{A}})$ must be equal or greater than the determinant of the original roughness matrix $\mathbf{A} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix}$, we use $\tau = \det(\mathbf{A}) = \alpha_x^2 \alpha_y^2$ to clamp $\det(\bar{\mathbf{A}})$.

B. Parameter Settings

While previous work [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan 2019] used $\sigma^2 = 0.25$, this paper uses $\sigma^2 = \frac{1}{2\pi}$ by assuming the pixel filter kernel is a Gaussian distribution whose peak is 1. For supersampling, we found this pixel filter kernel has less overblurring and visually closer to the box filter than $\sigma^2 = 0.25$ as shown in Figure 16. This parameter setting can also reduce the overfiltering error for our specular antialiasing.

C. Derivation of the Jacobian Matrix

Let ψ_x be an angle on the great circle passing through the halfvector $\mathbf{h}$ and normal $\mathbf{n}$, and ψ_y be an angle on the great circle passing through the halfvector $\mathbf{h}$ and $\frac{\mathbf{n}\times\mathbf{h}}{\|\mathbf{n}\times\mathbf{h}\|}$; then its Cartesian coordinate is given as

$$\begin{aligned} m_x &= \cos\psi_y \sin\psi_x, \\ m_y &= \sin\psi_y, \\ m_z &= \cos\psi_y \cos\psi_x. \end{aligned}$$

Thus, the Jacobian matrix of the transformation from $[\psi_x, \psi_y]$ to $[m_x, m_y]$ at $\psi_x = 0$ and $\psi_y = 0$ is given as

$$\mathbf{J}_{\mathrm{o}\to\perp^m} = \begin{bmatrix} \frac{\partial m_x}{\partial \psi_x} & \frac{\partial m_x}{\partial \psi_y} \\ \frac{\partial m_y}{\partial \psi_x} & \frac{\partial m_y}{\partial \psi_y} \end{bmatrix} = \begin{bmatrix} \cos\psi_y \cos\psi_x & -\sin\psi_y \sin\psi_x \\ 0 & \cos\psi_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The tangent-space halfvector can be represented using a polar-coordinate system $[\theta, \phi]$. Using this θ and this ϕ, the rotation from the local-space halfvector to tangent-space halfvector is

given by

$$\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & -\sin\phi & \sin\theta\cos\phi \\ \cos\theta\sin\phi & \cos\phi & \sin\theta\sin\phi \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix},$$

where $[m_x, m_y, m_z] = [0, 0, 1]$ (i.e., $\psi_x = 0$ and $\psi_y = 0$). Therefore, the Jacobian matrix of the orthographic projection is derived as

$$\begin{aligned} \mathbf{J}_{\circ\to\perp} = \mathbf{J}_{\perp^m\to\perp}\mathbf{J}_{\circ\to\perp^m} = \begin{bmatrix} \frac{\partial h_x}{\partial m_x} & \frac{\partial h_x}{\partial m_y} \\ \frac{\partial h_y}{\partial m_y} & \frac{\partial h_y}{\partial m_y} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & -\sin\phi \\ \cos\theta\sin\phi & \cos\phi \end{bmatrix} \\ = \frac{1}{\sqrt{1-h_z^2}} \begin{bmatrix} h_x h_z & -h_y \\ h_y h_z & h_x \end{bmatrix}. \end{aligned}$$

The slope of the halfvector is given as

$$h_x^{\parallel} = -\frac{h_x}{\sqrt{1-h_x^2-h_y^2}}, \qquad h_y^{\parallel} = -\frac{h_y}{\sqrt{1-h_x^2-h_y^2}}.$$

Therefore, the Jacobian matrix of the transformation from the projected unit disk to slope space is as follows:

$$\mathbf{J}_{\perp\to\parallel} = \begin{bmatrix} \frac{\partial h_x^{\parallel}}{\partial h_x} & \frac{\partial h_x^{\parallel}}{\partial h_y} \\ \frac{\partial h_y^{\parallel}}{\partial h_x} & \frac{\partial h_y^{\parallel}}{\partial h_y} \end{bmatrix} = -\frac{1}{h_z^3} \begin{bmatrix} 1-h_y^2 & h_x h_y \\ h_x h_y & 1-h_x^2 \end{bmatrix}.$$

Hence, the Jacobian matrix of the transformation from spherical space to slope space is obtained as

$$\mathbf{J}_{\circ\to\parallel} = \mathbf{J}_{\perp\to\parallel}\mathbf{J}_{\circ\to\perp} = -\frac{1}{h_z^2\sqrt{1-h_z^2}} \begin{bmatrix} h_x & -h_y h_z \\ h_y & h_x h_z \end{bmatrix}.$$

D. Optimization of Roughness Mapping Implementation

For our projected-space filtering, we optimize the implementation of roughness mapping by rewriting Equation (7) into the following equation:

$$\bar{\mathbf{A}} = \left(\bar{\mathbf{B}}^{-1} + \mathbf{I}\right)^{-1} = \frac{\operatorname{adj}\left(\frac{\operatorname{adj}(\bar{\mathbf{B}})}{\det(\bar{\mathbf{B}})} + \mathbf{I}\right)}{\det\left(\frac{\operatorname{adj}(\bar{\mathbf{B}})}{\det(\bar{\mathbf{B}})} + \mathbf{I}\right)} = \frac{\frac{\bar{\mathbf{B}}}{\det(\bar{\mathbf{B}})} + \mathbf{I}}{\det\left(\frac{\bar{\mathbf{B}}}{\det(\bar{\mathbf{B}})} + \mathbf{I}\right)}.$$

Using this equation, we eliminate the computation of adjugate matrices from our implementation (Listing 1).

Index of Supplemental Materials

The videos for our method can be found at `http://jcgt.org/published/0010/02/02/video.mp4`.

References

BECKMANN, P., AND SPIZZICHINO, A. 1963. *The Scattering of Electromagnetic Waves from Rough Surfaces*, vol. 4 of *Int. Ser. Monogr. Electromagn. Waves*. Pergamon Press. 34

CHEN, H. 2017. Toward film-like pixel quality in real-time games. In *GDC '17*. URL: https://www.gdcvault.com/play/1024273/Toward-Film-Like-Pixel-Quality. 32

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1, 7–24. URL: https://doi.org/10.1145/357290.357293. 33

HEITZ, E. 2014. Understanding the masking-shadowing function in microfacet-based BRDFs. *J. Comput. Graph. Tech. 3*, 2, 48–107. URL: http://jcgt.org/published/0003/02/03/. 34, 54

KAPLANYAN, A. S., HANIKA, J., AND DACHSBACHER, C. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph. 33*, 4. URL: https://doi.org/10.1145/2601097.2601108. 37

KAPLANYAN, A. S., HILL, S., PATNEY, A., AND LEFOHN, A. 2016. Filtering distributions of normals for shading antialiasing. In *HPG '16*, 151–162. URL: https://research.nvidia.com/publication/filtering-distributions-normals-shading-antialiasing. 32, 34, 35, 44, 45, 46, 47, 51, 53, 55

KAPLANYAN, A. S. 2016. Stable specular highlights. In *GDC '16*. URL: https://www.gdcvault.com/play/1023521/From-the-Lab-Bench-Real. 43, 45

KARIS, B. 2014. High-quality temporal supersampling. In *SIGGRAPH '14 Course: Advances in Real-Time Rendering in Games*. URL: http://advances.realtimerendering.com/s2014/. 32

SMITH, B. G. 1967. Geometrical shadowing of a random rough surface. *IEEE Trans. Antennas Propag. 15*, 5, 668–671. URL: https://doi.org/10.1109/TAP.1967.1138991. 34, 54

TOKUYOSHI, Y., AND KAPLANYAN, A. S. 2019. Improved geometric specular antialiasing. In *I3D '19*. URL: https://doi.org/10.1145/3306131.3317026. 32, 34, 55

UNITY TECHNOLOGIES, 2018. Unity Scriptable Render Pipeline. URL: https://github.com/Unity-Technologies/Graphics. 32, 44

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *EGSR '07*, 195–206. URL: https://doi.org/10.2312/EGWR/EGSR07/195-206. 34

Author Contact Information

Yusuke Tokuyoshi
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054-1549
yusuke.tokuyoshi@gmail.com
https://yusuketokuyoshi.com/

Anton S. Kaplanyan
Facebook Reality Labs
9845 Willows Rd.
Redmond, WA 98033
kaplanyan@fb.com
http://kaplanyan.com/

Y. Tokuyoshi and A. S. Kaplanyan, Stable Geometric Specular Antialiasing with Projected-Space NDF Filtering, *Journal of Computer Graphics Techniques (JCGT)*, vol. 10, no. 2, 31–58, 2021
http://jcgt.org/published/0010/02/02/

Received:	2020-03-14		
Recommended:	2020-11-28	Corresponding Editor:	Naty Hoffman
Published:	2021-05-20	Editor-in-Chief:	Marc Olano