

Improved Geometric Specular Antialiasing

Yusuke Tokuyoshi
SQUARE ENIX CO., LTD.
tokuyosh@square-enix.com

Anton S. Kaplanyan
Facebook Reality Labs
kaplanyan@fb.com

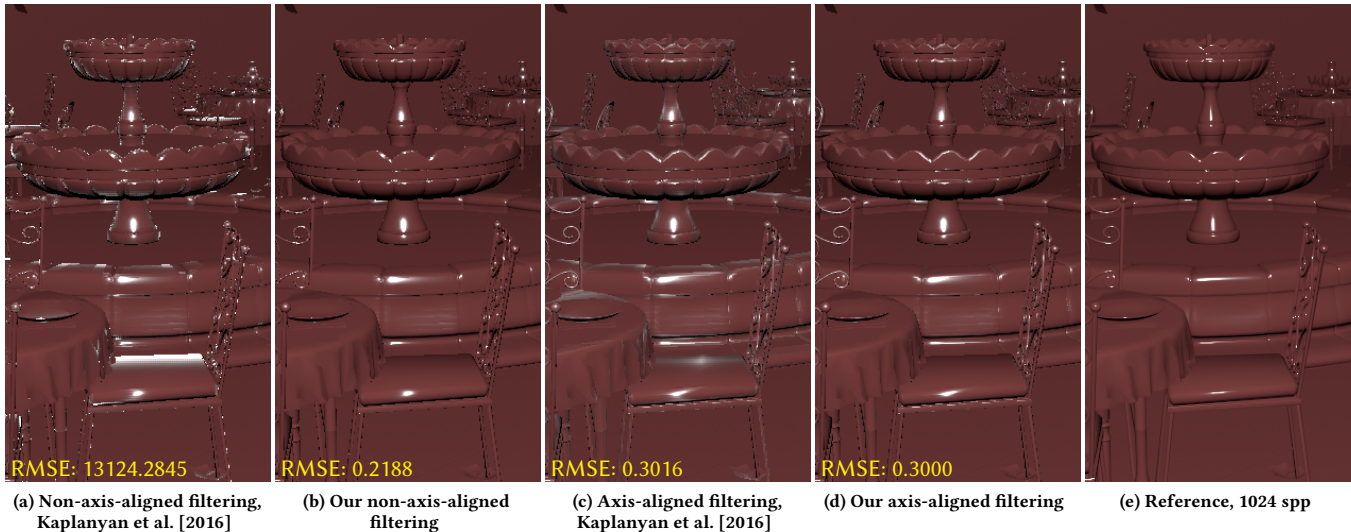


Figure 1: Improvements in geometric specular antialiasing for the GGX NDF with roughness $\alpha = 0.01$ on the SAN MIGUEL scene. Note the removed rims and more accurate specular highlights in our methods.

ABSTRACT

Shading filtering proposed by Kaplanyan et al. [2016] 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 more efficient kernel bandwidth that takes the angle of the halfvector into account. In addition, we optimize the calculation of an isotropic filter kernel used for deferred rendering by applying the proposed kernel bandwidth. As our implementation is simpler than the original method, it is easier to integrate in time-sensitive applications, such as game engines, while at the same time improving the filtering quality.

CCS CONCEPTS

• Computing methodologies → Rendering; Antialiasing.

KEYWORDS

specular antialiasing, real-time rendering, microfacet materials

ACM Reference Format:

Yusuke Tokuyoshi and Anton S. Kaplanyan. 2019. Improved Geometric Specular Antialiasing. In *Symposium on Interactive 3D Graphics and Games (I3D '19)*, May 21–23, 2019, Montreal, QC, Canada. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3306131.3317026>

1 INTRODUCTION

Rendering images in 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, 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 to miss 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 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 (SAA) 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] and 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 SAA method, while simultaneously decreasing the complexity of the code.

Our contributions are as follows:

- This work analyzes the increasing derivatives estimation error at grazing angles for geometric SAA (§3.1).
- To alleviate the above problem, we modify the kernel bandwidth by taking the angle of a halfvector into account for forward rendering (§3.2).
- We also present a simple roughness calculation for deferred rendering based on the proposed kernel bandwidth (§4).

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

2 BACKGROUND

2.1 Previous Work

Modern modeling of physically-based bidirectional reflectance distribution functions (BRDFs) uses a microfacet theory as the most common framework. In this model, light scattering during reflection off a rough surface is modeled using a distribution of microscopic specular facets, also called microfacets. The light scattering profile is then affected by the distribution of normals of such microfacets, called a normals distribution function (NDF). This microfacet BRDF model was introduced by Torrance [1967] and Cook [1982]. Heitz [2014] provides a recent survey on the microfacet theory.

Two common microfacet BRDFs used in renderers and game engines are the Beckmann distribution [1963] and the GGX distribution [Walter et al. 2007] (a.k.a., Trowbridge-Reitz [1975]). The Beckmann NDF models microfacet slopes as a bivariate Gaussian distribution of their tangents, also called slopes. The GGX NDF models the distribution of slopes induced by an ellipsoid and is typically characterized by a heavier tail of its specular highlights,

providing a closer match for real-world measured materials [Burley 2012; Walter et al. 2007].

For bump mapping, normal mapping, and displacement mapping, multiple filtering solutions were developed. Fournier [1992] introduced a combination of bump maps, shading surfaces, and surface BRDFs. Westin et al. [1992] studied filtering using complex multiscale materials. Becker et al. [1993] first proposed a seamless multiscale transition method between bump maps, displacement maps, and BRDF shading. They employed a multiscale hierarchy of BRDFs at different frequency levels. Kautz and Seidel [2000] filtered existing BRDF models using a shift-variant BRDF model and a non-linear basis. Toksvig [2005] introduced a real-time method for filtering normal maps by refitting a Gaussian NDF. Advanced normal map filtering methods includes a frequency-domain normal map filtering [Han et al. 2007] which uses multiple von Mises-Fisher modes to fit a multiscale complex representation of a high-resolution normal map into a mipmap pyramid that can be filtered in real-time. LEAN mapping [Olano and Baker 2010] stores the distribution of shading normals induced by high-resolution normal maps using a method of moments, where linearly filterable moments are stored in mip chains of textures. This approach was also extended to multiscale filtering of displacement maps [Dupuy et al. 2013].

There was a little work on efficient real-time methods for geometric specular antialiasing (SAA). The original work on SAA [Amanatides 1992] bandlimits the Phong BRDF by adjusting its exponent according to the pixel sampling frequency. Hill [2012] employed the Toksvig filter [2005] along with the derivative estimation of normals. Vlachos [2015] clamped the roughness with an empirical threshold using normal derivatives. Although these normal derivative-based approaches suppress specular aliasing, they induce the change of material appearance. Kaplanyan et al. [2016] builds on Amanatides [1992]’s method by projecting an estimation of the on-surface pixel footprint into the domain of microfacet slopes, where microfacet NDFs are defined. While this method provided a simple real-time implementation, it could suffer from an estimation error of the pixel footprint. To suppress this estimation error, Kaplanyan et al. [2016] also proposed an overfiltering approach using a biased axis-aligned filter kernel. Our solution improves on the recent progress in geometric SAA [Kaplanyan et al. 2016] by analyzing and reducing the main error of the method as well as proposing a more real-time friendly method.

2.2 Geometric Specular Antialiasing

The microfacet BRDF model 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, $\mathbf{h} = \frac{\mathbf{i} + \mathbf{o}}{\|\mathbf{i} + \mathbf{o}\|}$ is the halfvector, \mathbf{n} is the shading normal, $G_2(\mathbf{i}, \mathbf{o})$ is the masking-shadowing function, $F(\mathbf{i} \cdot \mathbf{h})$ is the Fresnel factor, and $D(\mathbf{h})$ is the NDF. See Table 1 for notation. For geometric SAA, 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

Table 1: Notation used throughout the paper.

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)
Σ	2×2 covariance matrix to represent the filter kernel
$\Delta \mathbf{h}_u^\parallel, \Delta \mathbf{h}_v^\parallel$	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 Σ

matrix Σ calculated for each pixel as follows:

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

where $\sigma = 0.5$ is the standard deviation 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 filtered NDF is also an anisotropic Beckmann NDF that uses the following 2×2 matrix as its roughness parameter:

$$\mathbf{A} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix} + 2\Sigma,$$

where α_x and α_y are the original Beckmann roughness parameters along the tangent and bitangent axes. In this paper, we refer to the above matrix as a *roughness matrix*, which is two times the covariance matrix by definition of the Beckmann NDF. It was also shown that this 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. Although microfacet BRDFs are usually used with axis-aligned anisotropy, these BRDFs are straightforward to generalize for handling a full roughness matrix for non-axis-aligned anisotropy [Heitz 2014]. For details on the generalized GGX microfacet BRDF, please refer to the supplemental document.

3 ERROR REDUCTION FOR SPECULAR ANTIALIASING

3.1 Error Analysis for Derivative Estimation

Grazing View Directions. 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/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 (Fig. 1a) 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

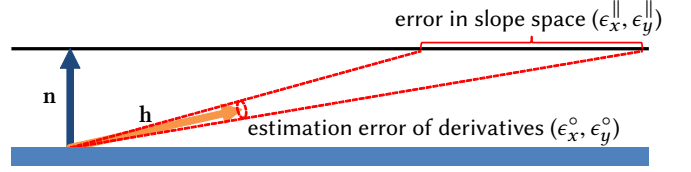


Figure 2: Estimation error of derivatives of the halfvector \mathbf{h} is increased in slope space for grazing \mathbf{h} . This error can become larger than the magnitude of the slope of \mathbf{h} , and thus inappropriate filtering is performed.

kernel. However, artifacts can still be noticeable for grazing angles as shown in Fig. 1c.

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

$$\begin{bmatrix} \epsilon_x^\parallel \\ \epsilon_y^\parallel \end{bmatrix} = J_{\mathbf{o} \rightarrow \parallel} \begin{bmatrix} \epsilon_x^\circ \\ \epsilon_y^\circ \end{bmatrix},$$

where ϵ_x^\parallel and ϵ_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 $J_{\mathbf{o} \rightarrow \parallel}$ of the transformation from spherical space to slope space is given by

$$J_{\mathbf{o} \rightarrow \parallel} = -\frac{1}{h_z^2 \sqrt{1-h_z^2}} \begin{bmatrix} h_x & -h_y h_z \\ h_y & h_x h_z \end{bmatrix},$$

where $[h_x, h_y, h_z]$ is the halfvector \mathbf{h} in tangent space (please refer to the supplemental document for derivation). The determinant of this Jacobian matrix is

$$\det(J_{\mathbf{o} \rightarrow \parallel}) = \frac{1}{h_z^3} \geq 1.$$

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 the supplemental document).

3.2 Proposed Error Reduction

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 (Fig. 3). For rough materials, filtering can be disabled all together, since it does not have an effect. Therefore, we reduce the error at grazing angles by using a narrower kernel bandwidth. To shrink the kernel bandwidth for grazing angles efficiently, this paper estimates the derivatives in a projected halfvector space instead of slope space

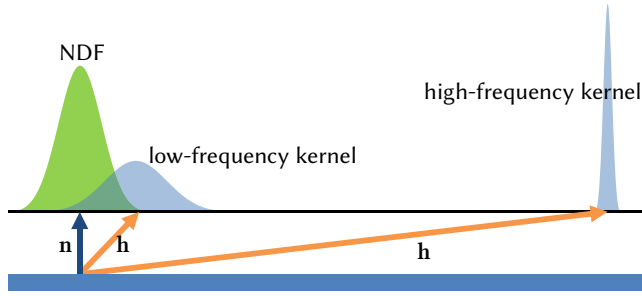


Figure 3: Since a grazing halfvector angle is unlikely to produce specular highlights (i.e., noticeable aliasing), we employ a higher-frequency filter kernel for a shallower halfvector angle.

(Fig. 4). This is done by the orthographic projection onto the h_x - h_y plane. Let Δh_u^\perp and Δh_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:

$$\Sigma = \sigma^2 \begin{bmatrix} \Delta h_u^\perp \\ \Delta h_v^\perp \end{bmatrix}^T \begin{bmatrix} \Delta h_u^\perp \\ \Delta h_v^\perp \end{bmatrix}.$$

As shown in Listing 1, this orthographic projection is simpler than the projection into slope space. The Jacobian matrix of this projection is given by

$$J_{o \rightarrow \perp} = \frac{1}{\sqrt{1-h_z^2}} \begin{bmatrix} h_x h_z & -h_y \\ h_y h_z & h_x \end{bmatrix}.$$

The determinant of this Jacobian matrix is

$$\det(J_{o \rightarrow \perp}) = h_z \leq 1.$$

Therefore, this projection reduces the error for grazing halfvectors. Fig. 1b and Fig. 1d show the rendering results using our kernel bandwidth. Our method significantly reduces artifacts without increasing the complexity of the shader code.

4 OPTIMIZATION FOR DEFERRED RENDERING

4.1 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 \bar{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:

$$\Sigma = \sigma^2 \begin{bmatrix} \Delta \bar{n}_u^\perp \\ \Delta \bar{n}_v^\perp \end{bmatrix}^T \begin{bmatrix} \Delta \bar{n}_u^\perp \\ \Delta \bar{n}_v^\perp \end{bmatrix},$$

where $\Delta \bar{n}_u^\perp$ and $\Delta \bar{n}_v^\perp$ are derivatives on the average normal \bar{n} on the projected unit disk. 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

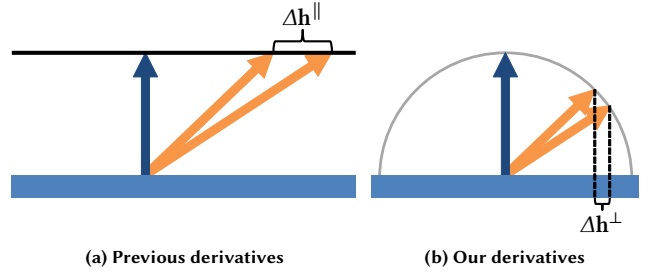


Figure 4: While the original method (a) estimates the derivatives in slope space, our method (b) estimates the derivatives on the projected unit disk to reduce the filter bandwidth.

Listing 1: Our derivative estimation for forward rendering (HLSL). The red code is removed from Kaplanyan et al. [2016]’s implementation.

```
float3 halfvector = normalize(viewDirection + lightDirection);
float3 halfvectorTS = mul(tangentFrame, halfvector);
float2 halfvector2D = halfvectorTS.xy; float2 halfvector2D = halfvectorTS.xy; // -abs(halfvectorTS.z);
float2 deltaU = ddx(halfvector2D), deltaV = ddy(halfvector2D);
```

kernel. In this section, we discuss alternative options for the case of a single scalar roughness. Using kernel bandwidth from Sect. 3.2, 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 2018]. The reason is that the computation cost of normal-based filtering is independent from 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 in these practical applications.

4.2 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 Σ). 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 Σ . Let λ_{\max} be the maximum eigenvalue. In this case, the squared roughness of the filtered isotropic NDF is given by

$$\bar{\alpha}^2 = \alpha^2 + \min(2\lambda_{\max}, \kappa), \tag{1}$$

where α is the original roughness parameter for the isotropic NDF, and $\kappa = 0.18$ is the clamping threshold used in the Kaplanyan et

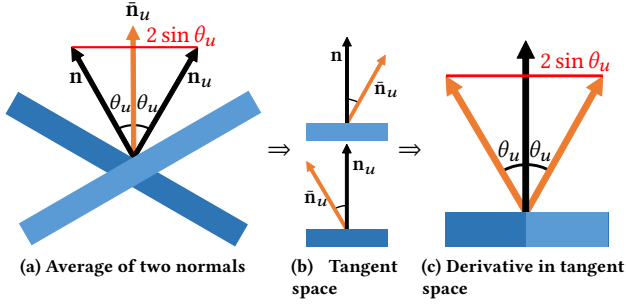


Figure 5: If the average normal $\bar{\mathbf{n}}_u$ is computed using two normals \mathbf{n} and \mathbf{n}_u of contiguous pixels (a), the length of the estimated derivative of the average normals in tangent space (c) is equal to the distance between \mathbf{n} and \mathbf{n}_u (i.e., length of the derivative of world-space normals).

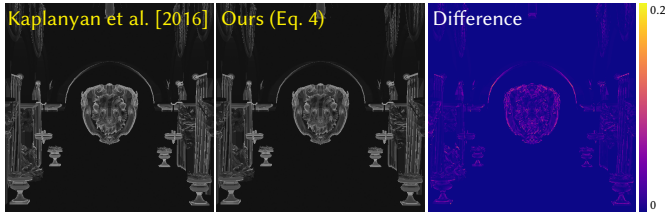


Figure 6: Visualization of roughness parameter $\tilde{\alpha}$ of the filtered isotropic NDF for deferred rendering.

al. [2016]’s original axis-aligned filtering to suppress the estimation error of derivatives. Although this eigenvalue λ_{\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.

4.3 Simple Conservative Filtering

Instead of the maximum eigenvalue λ_{\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 Σ as follows:

$$\lambda_{\max} \leq \lambda_{\min} + \lambda_{\max} = \text{tr}(\Sigma) = \sigma^2 \left(\|\Delta \bar{\mathbf{n}}_u^\perp\|^2 + \|\Delta \bar{\mathbf{n}}_v^\perp\|^2 \right). \quad (2)$$

4.4 Optimization for Norms of Derivatives

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

$$\bar{\mathbf{n}}_u = \frac{\mathbf{n} + \mathbf{n}_u}{\|\mathbf{n} + \mathbf{n}_u\|}, \quad \bar{\mathbf{n}}_v = \frac{\mathbf{n} + \mathbf{n}_v}{\|\mathbf{n} + \mathbf{n}_v\|},$$

where \mathbf{n}_u and \mathbf{n}_v are normals at the neighboring pixels for each screen axis. For this case, $\bar{\mathbf{n}}_u$ is on the great circle defined by \mathbf{n} and \mathbf{n}_u , and the angle between \mathbf{n} and $\bar{\mathbf{n}}_u$ is equal to the angle between \mathbf{n}_u and $\bar{\mathbf{n}}_u$. As shown in Fig. 5, let this angle be θ_u , then the norm of the derivative of $\bar{\mathbf{n}}_u^\perp$ is given by

$$\|\Delta \bar{\mathbf{n}}_u^\perp\| = 2 \sin \theta_u = \|\mathbf{n} - \mathbf{n}_u\|.$$

Listing 2: Our roughness calculation using conservative filter for deferred rendering (HLSL). The red code is removed for less conservative filtering.

```
float3 dndu = ddx(normal), dndv = ddy(normal);
float variance = SIGMA2 * (dot(dndu, dndu) + dot(dndv, dndv));
float kernelRoughness2 = min(2.0 * variance, KAPPA);
float filteredRoughness2 = saturate(roughness2 + kernelRoughness2);
```

Listing 3: Previous roughness calculation using the width of the rectangular kernel for deferred rendering (HLSL).

```
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 deltaU = ddx(avgNormal2D), deltaV = ddy(avgNormal2D);
float2 boundingRectangle = abs(deltaU) + abs(deltaV);
float maxWidth = max(boundingRectangle.x, boundingRectangle.y);
float variance = SIGMA2 * maxWidth * maxWidth;
float kernelRoughness2 = min(2.0 * variance, KAPPA);
float filteredRoughness2 = saturate(roughness2 + kernelRoughness2);
```

The same relation is yielded for \mathbf{n} , \mathbf{n}_v , and $\bar{\mathbf{n}}_v$. 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

$$\|\Delta \bar{\mathbf{n}}_u^\perp\| = \|\Delta \mathbf{n}_u\|, \quad \|\Delta \bar{\mathbf{n}}_v^\perp\| = \|\Delta \mathbf{n}_v\|. \quad (3)$$

4.5 Optimized Conservative Filtering

By substituting Eq. 3 to Eq. 2, 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

$$\tilde{\alpha}^2 = \alpha^2 + \min \left(2\sigma^2 \left(\|\Delta \mathbf{n}_u\|^2 + \|\Delta \mathbf{n}_v\|^2 \right), \kappa \right). \quad (4)$$

Since this calculation uses world-space normals, the computation of the average normal and transformation into tangent space are unnecessary. Our implementation (Listing 2) is simpler than computing the rectangular kernel-based maximum roughness (Listing 3). In addition, since this roughness calculation is independent from tangent vectors, it performs robustly even if objects do not have valid tangent vectors. The visualization of the filtered roughness parameter is shown Fig. 6. For deferred rendering, there are no large differences between our method and Kaplanyan et al. [2016], while the proposed implementation is simpler.

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

$$\begin{aligned} \tilde{\alpha}^2 &= \alpha^2 + \min \left(2 \left(\frac{\lambda_{\min} + \lambda_{\max}}{2} \right), \kappa \right) \\ &= \alpha^2 + \min \left(\sigma^2 \left(\|\Delta \mathbf{n}_u\|^2 + \|\Delta \mathbf{n}_v\|^2 \right), \kappa \right). \end{aligned} \quad (5)$$

Table 2: Computation time of forward shading at 8K resolution (ms).

	w/o SAA	Non-axis-aligned filtering		Axis-aligned filtering		Normal-based isotropic filtering			
		Previous	Ours	Previous	Ours	Previous	Max (Eq. 1)	Sum (Eq. 4)	Avg (Eq. 5)
SPONZA	1.80	2.34	2.31	2.35	2.30	2.35	2.40	1.90	1.88
BISTRO	2.06	2.60	2.57	2.61	2.56	2.58	2.62	2.16	2.17
SAN MIGUEL	3.65	4.16	4.12	4.15	4.12	4.16	4.19	3.74	3.74



Figure 7: Reference images in our experiments (1024 spp).

5 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 AMD Radeon™ RX Vega 56 GPU. The image quality is evaluated with the root-mean-squared error (RMSE) metric. In our experiments, reference images shown in Fig. 7 are rendered using 1024 samples/pixel (spp). The RMSE for each scene using 1 spp is 1.1045 (SPONZA), 1.0881 (BISTRO), and 1.0663 (SAN MIGUEL), respectively.

Forward Rendering. Listing 1 shows the necessary code changes and Fig. 8 demonstrates the improvements for forward rendering. Our method reduces the error for non-axis-aligned filtering significantly. In this experiment, our non-axis-aligned filtering is the highest quality in terms of the RMSE metric, while some pixels can still flicker in animation (please see the supplemental video). For such 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.

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 2, while the previous algorithm is provided in Listing 3. The quality comparison of these methods is shown in Fig. 9. NDF filtering using the maximum eigenvalue (Eq. 1) has the smallest error in the constraint of conservative filtering in theory. The proposed conservative filtering (Eq. 4) produces slightly smaller error than the previous method, while our implementation is much simpler than the previous method and the above optimal approach. These techniques are conservative to avoid underfiltering, but they induce overfiltering instead. Our less conservative filtering (Eq. 5) reduces the error by balancing overfiltering and underfiltering. While less conservative filtering can produce temporal 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 computation time for forward shading at 8K resolution. As described in Sect. 4.1, since normal-based isotropic NDF 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, the computation time of our method is almost the same as or slightly smaller than the previous method, because our derivative estimation is simpler than the previous method. For normal-based isotropic NDF filtering, conservative filtering using the maximum eigenvalue is slightly more expensive than the previous method. On the other hand, our simple filtering techniques using the sum of eigenvalues and average of eigenvalues are faster than the previous method, while they produce less error. The performance improvement of our simplification is effective when shaders are ALU bound.

6 LIMITATIONS

Our method is built upon Kaplanyan et al. [2016]’s work and inherits the usual limitations of this previous work. The method heavily relies on high quality shading normals and addresses only the aliasing caused by specular highlights. Aliasing caused by geometric discontinuities cannot be handled by the method. Real-time approximation of a pixel footprint introduces bias. The filtering of the GGX NDF is approximated by assuming the Beckmann NDF, therefore, GGX highlights can be overblurred due to this approximation. Unlike the previous method, the proposed kernel produces underfiltering for grazing halfvectors. However, it is usually not a problem because aliasing is small for grazing halfvectors.

7 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 employs a higher-frequency filter kernel for a shallower halfvector angle. This is implemented by estimating derivatives of a projected halfvector instead of a slope space halfvector. In addition, we presented an 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 SAA.

ACKNOWLEDGMENTS

The polygon models are courtesy of F. Meinel for SPONZA, the Amazon Lumberyard team for BISTRO, and Guillermo M. Leal Llaguno for SAN MIGUEL. The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

J. Amanatides. 1992. Algorithms for the Detection and Elimination of Specular Aliasing. In *Graphics Interface '92*. 86–93.

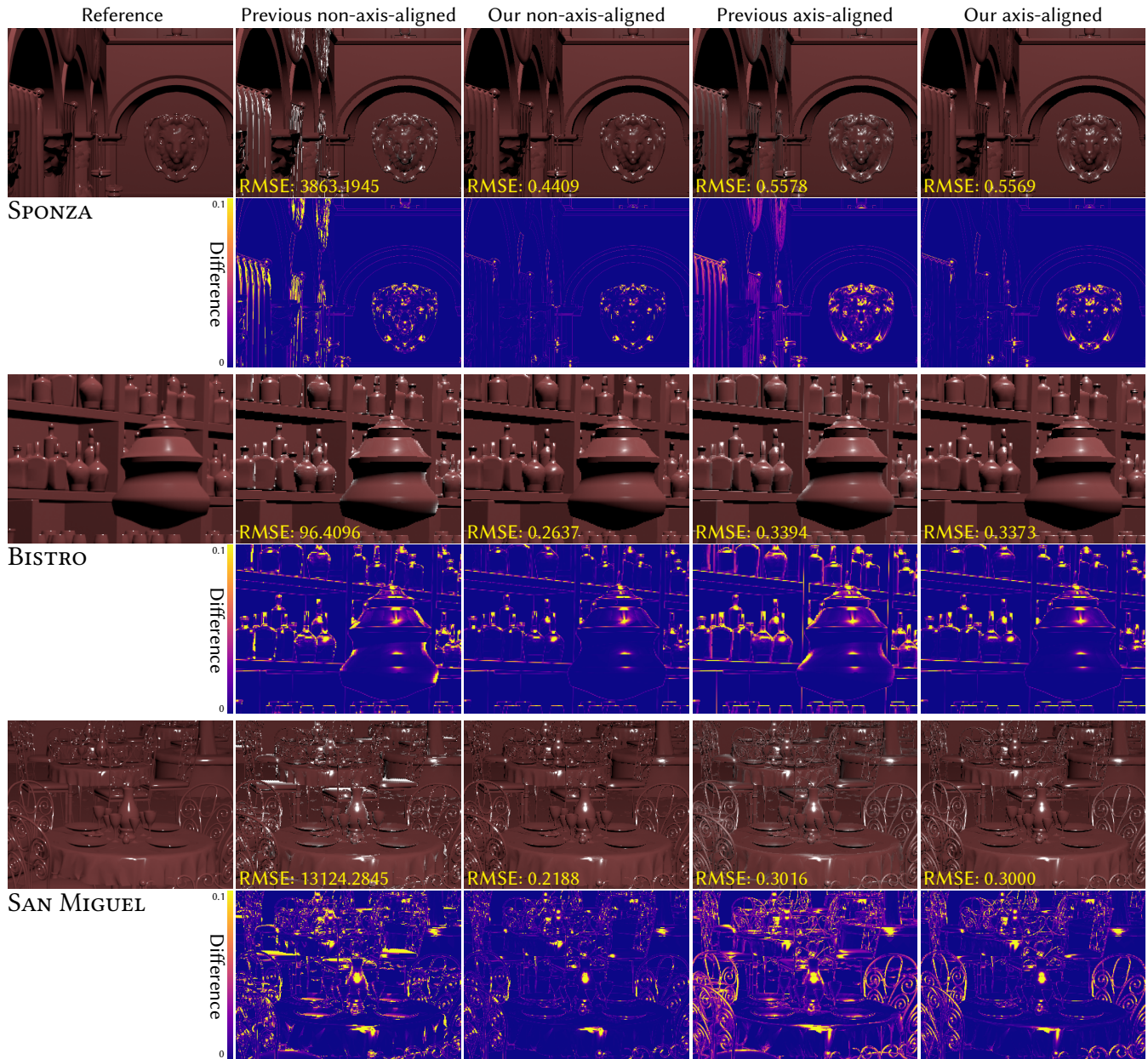


Figure 8: Quality comparison of NDF filtering for forward rendering. Images are closeups of rendering results and the difference from the reference. Our method reduces the error at grazing angles both for non-axis-aligned and axis-aligned filtering.

B. G. Becker and N. L. Max. 1993. Smooth Transitions Between Bump Rendering Algorithms. In *SIGGRAPH '93*. 183–190.

P. Beckmann and A. Spizzichino. 1963. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon Press.

B. Burley. 2012. Physically-Based Shading at Disney. In *SIGGRAPH '12 Course: Practical Physically-based Shading in Film and Game Production*. 10:1–10:7.

H. Chen. 2017. Toward Film-Like Pixel Quality in Real-Time Games. In *GDC '17*.

R. L. Cook and K. E. Torrance. 1982. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1, 1 (1982), 7–24.

J. Dupuy, E. Heitz, J.-C. Iehl, P. Poulin, F. Neyret, and V. Ostromoukhov. 2013. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Trans. Graph.* 32, 6 (2013), 211:1–211:11.

A. Fournier. 1992. Normal Distribution Functions and Multiple Surfaces. In *Graphics Interface '92 Workshop on Local Illumination*. 45–52.

C. Han, B. Sun, R. Ramamoorthi, and E. Grinspun. 2007. Frequency Domain Normal Map Filtering. *ACM Trans. Graph.* 26, 3, Article 28 (2007).

E. Heitz. 2014. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *J. Comput. Graph. Tech.* 3, 2 (2014), 48–107.

S. Hill and D. Baker. 2012. Rock-Solid Shading: Image Stability Without Sacrificing Detail. In *SIGGRAPH '12 Course: Advances in Real-Time Rendering in Games*.

A. S. Kaplanyan. 2016. Stable Specular Highlights. In *GDC '16*.

A. S. Kaplanyan, J. Hanika, and C. Dachsbacher. 2014. The Natural-constraint Representation of the Path Space for Efficient Light Transport Simulation. *ACM Trans. Graph.* 33, 4 (2014), 102:1–102:13.

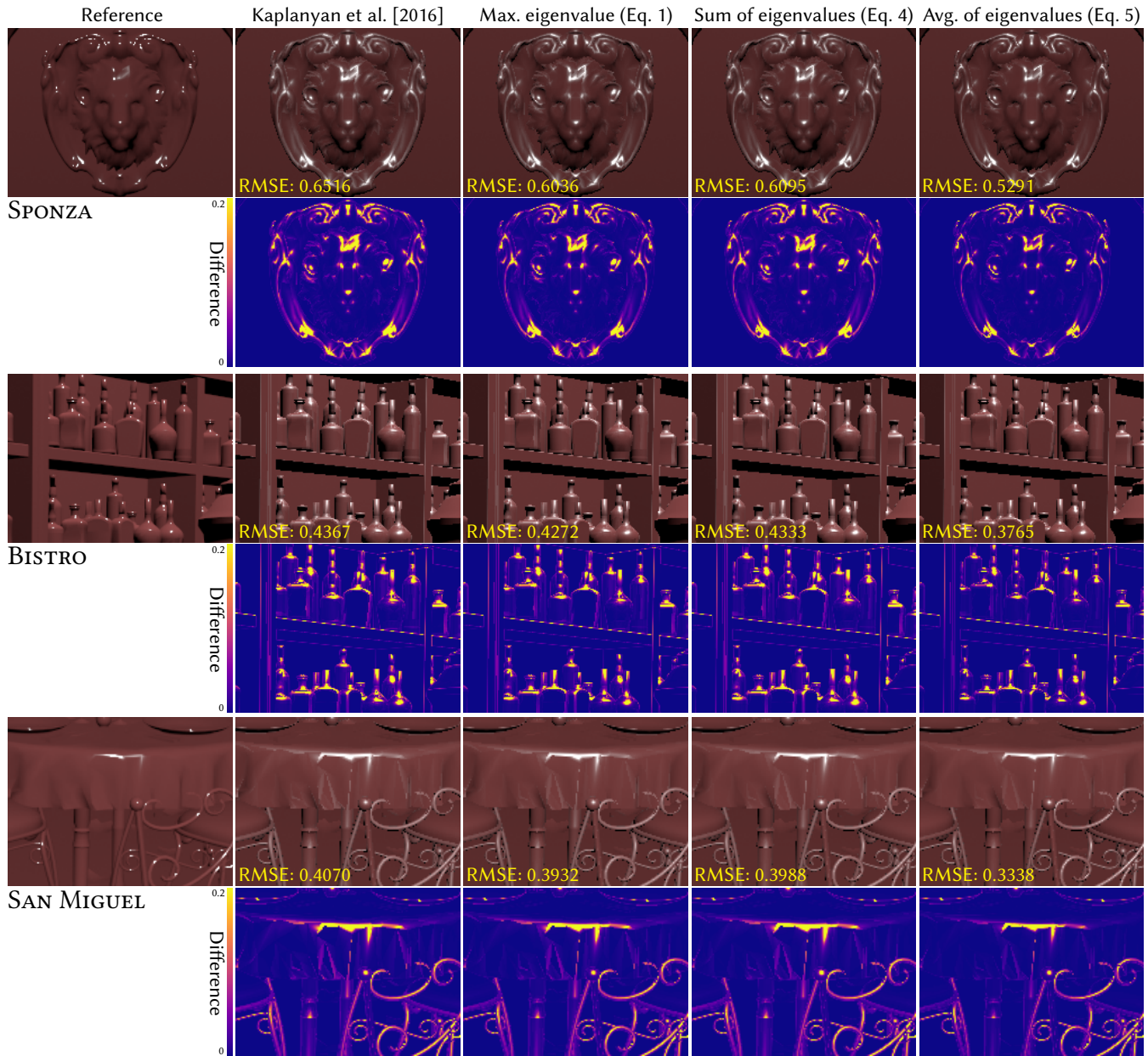


Figure 9: Quality comparison of isotropic NDF filtering for deferred rendering (closeups). Our less conservative filtering using the average of eigenvalues (rightmost) produces the lowest RMSE in this experiment, while its implementation is the simplest.

A. S. Kaplanyan, S. Hill, A. Patney, and A. Lefohn. 2016. Filtering Distributions of Normals for Shading Antialiasing. In *HPG '16*. 151–162.

B. Karis. 2014. High-Quality Temporal Supersampling. In *SIGGRAPH '14 Course: Advances in Real-Time Rendering in Games*. 12:1–12:11.

J. Kautz and H.-P. Seidel. 2000. Towards Interactive Bump Mapping with Anisotropic Shift-variant BRDFs. In *HWWS '00*. 51–58.

M. Olano and D. Baker. 2010. LEAN Mapping. In *I3D '10*. 181–188.

M. Toksvig. 2005. Mipmapping Normal Maps. *J. Graph. GPU, and Game Tools* 10, 3 (2005), 65–71.

K. E. Torrance and E. M. Sparrow. 1967. Theory for Off-Specular Reflection From Roughened Surfaces. *J. Opt. Soc. Am.* 57, 9 (1967), 1105–1114.

T. S. Trowbridge and K. P. Reitz. 1975. Average Irregularity Representation of a Rough Surface for Ray Reflection. *J. Opt. Soc. Am.* 65, 5 (1975), 531–536.

Unity. 2018. Scriptable Render Pipeline. <https://github.com/Unity-Technologies/ScriptableRenderPipeline>

A. Vlachos. 2015. Advanced VR Rendering. In *GDC '15*.

B. Walter, S. Marschner, H. Li, and K. Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *EGSR '07*. 195–206.

S. H. Westin, J. R. Arvo, and K. E. Torrance. 1992. Predicting Reflectance Functions from Complex Surfaces. *SIGGRAPH Comput. Graph.* 26, 2 (1992), 255–264.