Error Reduction and Simplification for Shading Anti-Aliasing

Yusuke Tokuyoshi*
Square Enix Co., Ltd.

1 Introduction

The shading anti-aliasing technique [Kaplanyan et al. 2016] is a simple solution for specular aliasing. This report improves the quality and performance of this technique without increasing the complexity of the shader code. The shading anti-aliasing technique filters a normal distribution function (NDF, i.e., distribution of microfacet normals) in the slope domain. However, for real-time rendering, this approach can produce noticeable artifacts because of an estimation error of derivatives (e.g., error of *ddx/ddy* instructions in HLSL). For forward rendering, this estimation error is increased significantly by projecting a halfvector into slope space (Fig. 1). To reduce the error, this report introduces an adaptive kernel bandwidth which takes the angle of the halfvector into account. In addition, we also simplify the calculation of an isotropic filter kernel for deferred rendering by using the proposed kernel bandwidth. Our implementation is simpler than the original method, therefore it is more suitable for time-sensitive applications.

Our contributions are as follows:

- This report clarifies that a derivative estimation error is increased in slope space for shading anti-aliasing (§3.1).
- To alleviate the above problem, we modify the kernel bandwidth taking the angle of a halfvector into account for forward rendering (§3.2).
- We present a simple roughness calculation for deferred rendering based on the proposed kernel bandwidth (§4).

2 Shading Anti-Aliasing

The microfacet BRDF model [Cook and Torrance 1982] is defined as

$$f(\mathbf{i},\mathbf{o}) = \frac{G_2(\mathbf{i},\mathbf{o})D(\mathbf{h})F(\mathbf{h}\cdot\mathbf{o})}{4|\mathbf{i}\cdot\mathbf{n}||\mathbf{o}\cdot\mathbf{n}|}, \tag{1}$$

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 surface normal, $G_2(\mathbf{i},\mathbf{o})$ is the masking-shadowing function, and $F(\mathbf{h}\cdot\mathbf{o})$ is the Fresnel factor, and $D(\mathbf{h})$ is the NDF. For shading anti-aliasing, the Beckmann NDF [Beckmann and Spizzichino 1963] is assumed and then it is filtered using an anisotropic Gaussian kernel in the slope domain. The kernel bandwidth is given as a covariance matrix $\mathbf{\Sigma}$ calculated for each pixel as follows:

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

where $\sigma^2 = 0.25$ is the variance of the filter kernel in screen space, and $\Delta\mathbf{h}_u^{\parallel}$ and $\Delta\mathbf{h}_v^{\parallel}$ are the derivatives of the halfvector in slope space for each pixel. Since the Beckmann NDF is a Gaussian distribution in slope space, this filtering is a convolution of two Gaussian distributions. Hence, the filtered NDF is also an anisotropic Beckmann NDF which uses the following 2×2 matrix as a roughness parameter:

$$\mathbf{A} = \begin{bmatrix} \alpha_x^2 & 0 \\ 0 & \alpha_y^2 \end{bmatrix} + 2\mathbf{\Sigma}, \tag{3}$$

*e-mail:tokuyosh@square-enix.com

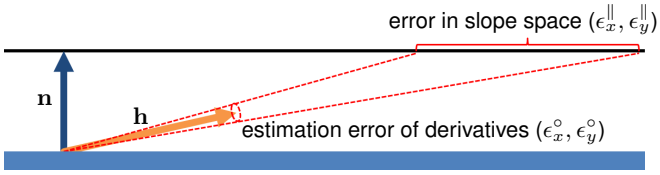


Figure 1: *Estimation error of derivatives of the halfvector* $\mathbf{h}$ *is increased in slope space for shallow* $\mathbf{h}$*. This error can be larger than the magnitude of the slope of* $\mathbf{h}$*, and thus inappropriate filtering is performed.*

where α_x and α_y are the original Beckmann roughness parameters for the tangent axis and bitangent axis. In this report, we describe the above matrix as a *roughness matrix* which is two times of a covariance matrix. This roughness matrix can also be used approximately for the GGX NDF [Trowbridge and Reitz 1975; Walter et al. 2007]. Although microfacet BRDFs are usually formulated using axis-aligned anisotropy, these BRDFs can be generalized using this roughness matrix for non-axis-aligned anisotropy [Heitz 2014]. For the detail of the generalized GGX microfacet BRDF used in this report, please refer to Appendix A.

3 Error Reduction for Shading Anti-aliasing

3.1 Error of derivative estimation

Shallow view directions. For real-time computer graphics APIs (e.g., DirectX® and OpenGL®), derivatives are roughly estimated in a pixel shader using intrinsic functions (e.g., ddx/ddy instructions in HLSL). These intrinsics compute the difference between values of two contiguous pixels in the 2×2 shading pixel quad. However, this rough estimation produces a numerical error (Fig. 2a) especially for a shallow grazing angle of the view direction $\mathbf{o}$. To suppress artifacts caused by this error, Kaplanyan et al. [2016] proposed a biased axis-aligned rectangular filtering technique. They also clamped the bandwidth of their rectangular kernel. However, artifacts can still be noticeable for shallow grazing angles as shown in Fig. 3a.

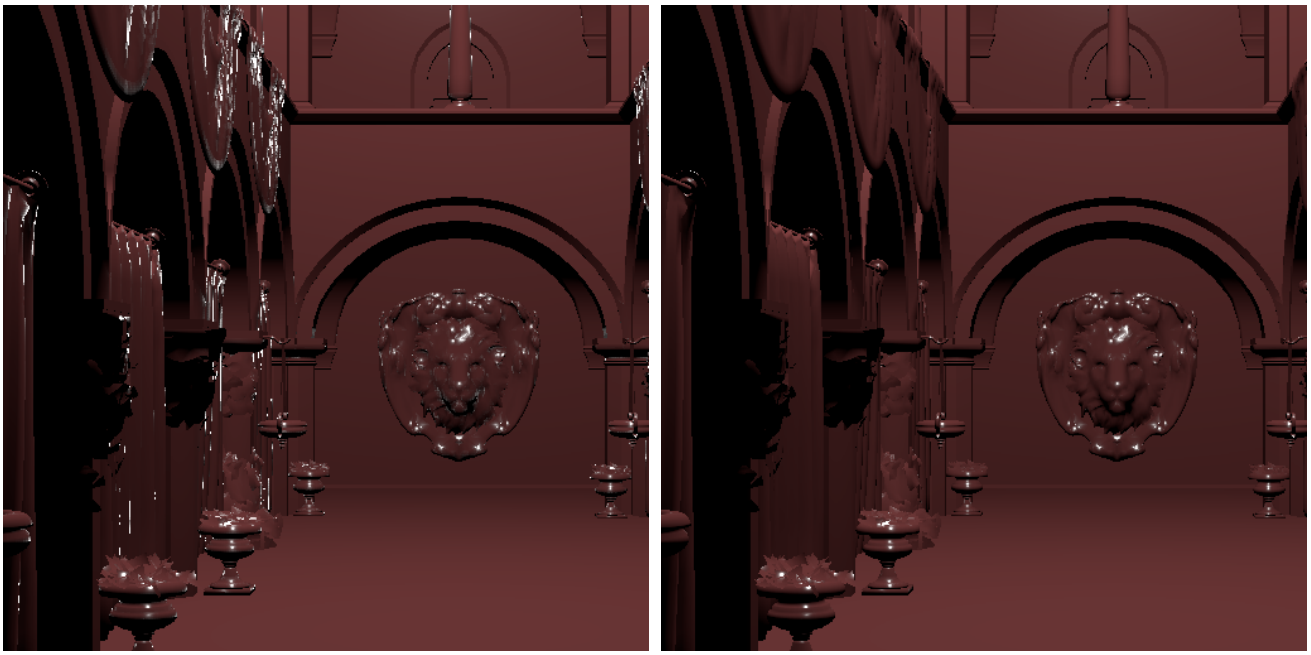
Shallow halfvectors. In this report, we show that the above estimation error is intensely increased by projecting the halfvector $\mathbf{h}$ into slope space. This increase of the error is represented using the Jacobian matrix of the projection as follows:

$$\begin{bmatrix} \epsilon_x^{\parallel} \\ \epsilon_y^{\parallel} \end{bmatrix} = J_{\circ\to\parallel} \begin{bmatrix} \epsilon_x^{\circ} \\ \epsilon_y^{\circ} \end{bmatrix}, \tag{4}$$

where $\epsilon_x^{\parallel}$ and $\epsilon_x^{\parallel}$ 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 $\frac{\mathbf{n}\times\mathbf{h}}{\|\mathbf{n}\times\mathbf{h}\|}$. $J_{\circ\to\parallel}$ is the Jacobian matrix of the transformation from spherical space to slope space given by:

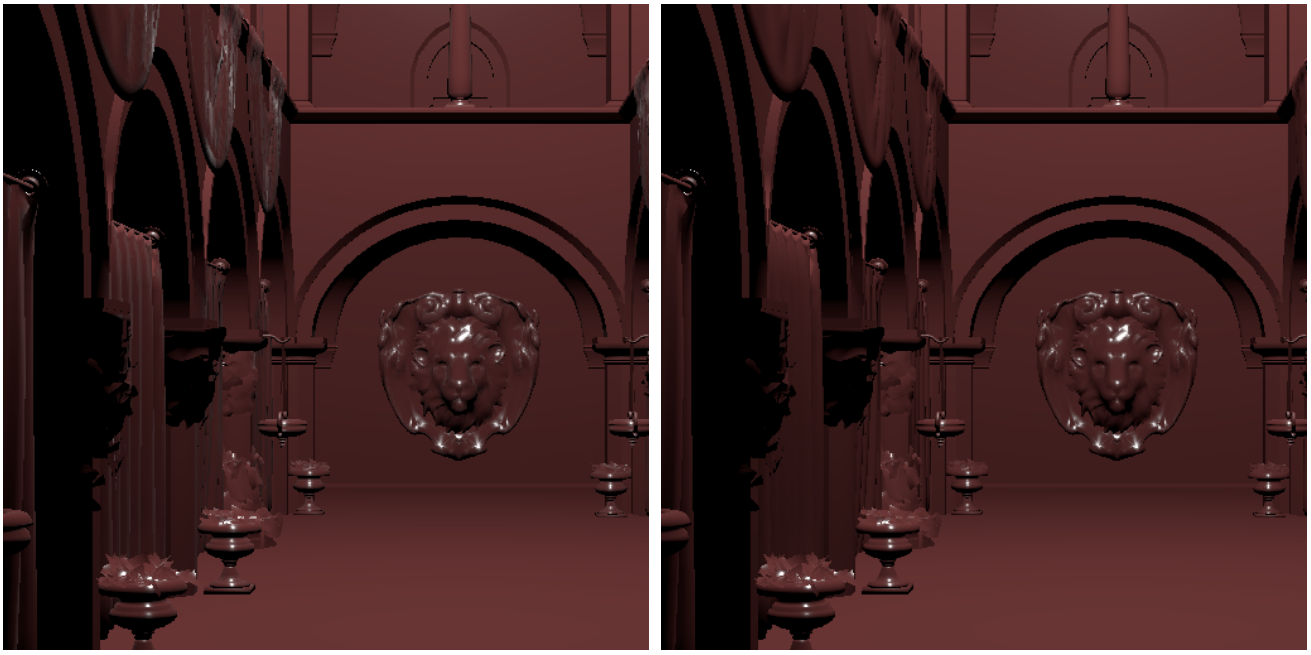
$$J_{\circ\to\parallel} = -\frac{1}{h_z^2\sqrt{1-h_z^2}} \begin{bmatrix} h_x & -h_y h_z \\ h_y & h_x h_z \end{bmatrix}, \tag{5}$$

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



(a) *Kaplanyan et al. 2016* **(b)** *Ours*

Figure 2: *Non-axis-aligned filtering using covariance matrix* $\mathbf{\Sigma}$ *for the GGX NDF.*



(a) *Kaplanyan et al. 2016* **(b)** *Ours*

Figure 3: *Biased axis-aligned rectangular filtering for the GGX NDF.*

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

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

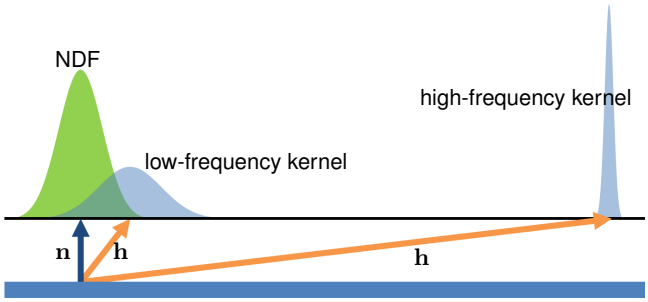
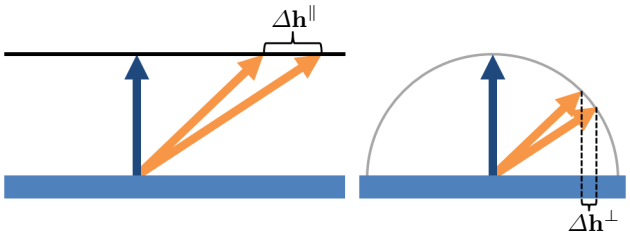


Figure 4: *Since a shallow halfvector angle does not produce highlights (i.e., noticeable aliasing), we employ a higher-frequency filter kernel for a shallower halfvector angle.*



(a) *Previous derivatives* **(b)** *Our derivatives*

Figure 5: *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 kernel bandwidth.*

Jacobian matrix is

$$\det\left(J_{\circ\to\parallel}\right) = \frac{1}{h_z^3} \geq 1. \tag{6}$$

The magnitude of the error in slope space can be larger than the magnitude of the slope of the halfvector for small h_z. Hence, significant artifacts are induced for shallow halfvectors due to inappropriate filtering. These artifacts are noticeable especially for the GGX NDF, because the GGX distribution has a stronger tail than the Beckmann distribution.

3.2 Our approach

Artifacts are produced when halfvectors are shallow grazing angles. However, NDF filtering is often unnecessary for such shallow halfvectors because they do not produce highlights. Therefore, we reduce the kernel bandwidth according to the angle of the halfvector (Fig. 4). This is easily implemented by estimating the derivatives in a different space instead of slope space. For this derivative estimation, this report employs the orthographic projection onto the h_x-h_y-plane (Fig. 5). Let $\Delta\mathbf{h}_u^{\perp}$ and $\Delta\mathbf{h}_v^{\perp}$ be the derivatives of $[h_x, h_y]$: then our filter kernel is given by the following covariance matrix:

$$\boxed{\mathbf{\Sigma} = \sigma^2 \begin{bmatrix}\Delta\mathbf{h}_u^{\perp}\\ \Delta\mathbf{h}_v^{\perp}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\Delta\mathbf{h}_u^{\perp}\\ \Delta\mathbf{h}_v^{\perp}\end{bmatrix}}. \tag{7}$$

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

$$J_{\circ\to\perp} = \frac{1}{\sqrt{1-h_z^2}} \begin{bmatrix} h_x h_z & -h_y \\ h_y h_z & h_x \end{bmatrix}. \tag{8}$$

The determinant of this Jacobian matrix is

$$\det\left(J_{\circ\to\perp}\right) = h_z \leq 1. \tag{9}$$

Thus, this projection reduces the error for shallow halfvectors. Fig. 2b and Fig. 3b shows the rendering results using our kernel bandwidth. Our method significantly reduces artifacts without increasing the complexity of the shader code.

4 Simplification for Deferred Rendering

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. Therefore, an average normal $\bar{\mathbf{n}}$ within the shading quad is used instead of the halfvector $\mathbf{h}$ for derivative estimation by assuming the worst case for a distant light source and distant eye position. For a compact G-buffer, since a single scalar roughness parameter is often required, Kaplanyan et al. [2016] used the maximum roughness of their rectangular filter kernel. In this section, we discuss other options for such single scalar roughness. Using our kernel bandwidth presented in 3.2, this report derives a simple isotropic filter kernel.

Constraint for isotropic NDF filtering. The bandwidth of the isotropic kernel must be wider than the original anisotropic kernel (represented with the covariance matrix $\mathbf{\Sigma}$) to completely remove specular aliasing. The maximum roughness of Kaplanyan et al.'s rectangular kernel satisfies this constraint. On the other hand, this kernel bandwidth should be small to reduce overfiltering. Within this constraint and objective, the optimal kernel bandwidth is obtained as the maximum eigenvalue of the covariance matrix $\mathbf{\Sigma}$. Let λ be the maximum eigenvalue: then the squared roughness of the filtered isotropic NDF is given by

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

where α is the original roughness parameter for the isotropic NDF, and $\delta_{\max} = 0.18$ is the clamping threshold used in the Kaplanyan [2016]'s rectangular filtering to suppress the estimation error of derivatives. Although this eigenvalue λ can be calculated analytically, it is slightly more expensive than the rectangular kernel-based approach. This report proposes another option which satisfies the above constraint.

Our isotropic filter kennel bandwidth. We employ the sum of squared norms of derivatives which is equal or greater than the maximum eigenvalue as follows:

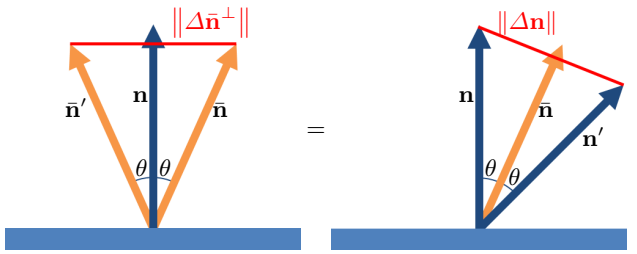
$$\lambda \leq \sigma^2 \left(\left\| \Delta\bar{\mathbf{n}}_u^{\perp} \right\|^2 + \left\| \Delta\bar{\mathbf{n}}_v^{\perp} \right\|^2 \right), \tag{11}$$

where $\Delta\bar{\mathbf{n}}_u^{\perp}$ and $\Delta\bar{\mathbf{n}}_v^{\perp}$ are derivatives on the average normal $\bar{\mathbf{n}}$ on the projected unit disk. While Kaplanyan et al. [2016] computed $\bar{\mathbf{n}}$ using the average within the shading quad, in this report we employ the average of two contiguous pixels for each screen axis. For this case, $\left\|\Delta\bar{\mathbf{n}}_u^{\perp}\right\|$ and $\left\|\Delta\bar{\mathbf{n}}_v^{\perp}\right\|$ are respectively equal to $\|\Delta\mathbf{n}_u\|$ and $\|\Delta\mathbf{n}_v\|$ which are lengths of derivatives of world-space normals as shown in Fig. 6. Thus we obtain the following equation:

$$\sigma^2 \left(\left\| \Delta\bar{\mathbf{n}}_u^{\perp} \right\|^2 + \left\| \Delta\bar{\mathbf{n}}_v^{\perp} \right\|^2 \right) = \sigma^2 \left(\|\Delta\mathbf{n}_u\|^2 + \|\Delta\mathbf{n}_v\|^2 \right). \tag{12}$$

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

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



(a) *Derivative of average normals in tangent space* **(b)** *Derivative of normals in world space*

Figure 6: *If the average normal $\bar{\mathbf{n}}$ is computed using two normals $\mathbf{n}$ and $\mathbf{n}'$ of contiguous pixels, the length of the estimated derivative of the average normals in tangent space (a) is equal to the distance between $\mathbf{n}$ and $\mathbf{n}'$ (i.e., length of the derivative of world-space normals) (b).*

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 rendering results and visualization of the filtered roughness parameter are shown in Fig. 7 and Fig. 8, respectively. For deferred rendering, there are no large visual differences between our method and Kaplanyan et al. [2016], while the proposed implementation is simpler.

5 Conclusions

In this report 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 report employs a higher-frequency filter kernel for a shallower halfvector angle. This is implemented by estimating derivatives on the projected unit disk instead of slope space. In addition, this report also presents a simpler isotropic NDF filtering technique for deferred rendering based on this derivative estimation. Hence, our method does not only reduce the error, but also simplifies the shader code for shading anti-aliasing.

Acknowledgements

The polygon models are courtesy of M. Dabrovic and F. Meinl. The author would like to thank Anton S. Kaplanyan for valuable comments.

A Non-Axis-Aligned Anisotropic BRDF

Masking-shadowing function. The Smith [1967] masking function is defined as $G_1(\mathbf{o},\mathbf{h}) = \frac{\chi^+(\mathbf{o}\cdot\mathbf{h})}{1+\Lambda(\mathbf{o})}$, where $\chi^+(\mathbf{o}\cdot\mathbf{h})$ is the Heaviside function: 1 if $\mathbf{o}\cdot\mathbf{h} > 0$ otherwise 0. $\Lambda(\mathbf{o})$ is a function which depends on the NDF model. The height-correlated masking-shadowing function [Heitz 2014] is given as

$$G_2(\mathbf{i},\mathbf{o}) = \frac{\chi^+(\mathbf{i}\cdot\mathbf{h})\,\chi^+(\mathbf{o}\cdot\mathbf{h})}{1+\Lambda(\mathbf{i})+\Lambda(\mathbf{o})}. \tag{14}$$

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

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

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\alpha_x\alpha_y\left(\frac{h_x^2}{\alpha_x^2}+\frac{h_y^2}{\alpha_y^2}+h_z^2\right)^2}. \tag{15}$$

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

$$\Lambda(\mathbf{o}) = -0.5 + \frac{\sqrt{\alpha_x^2 o_x^2 + \alpha_y^2 o_y^2 + o_z^2}}{2|o_z|}, \tag{16}$$

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

Non-axis-aligned anisotropic GGX BRDF. For shading anti-aliasing, we use the 2×2 roughness matrix $\mathbf{A}$ instead of α_x and α_y. The anisotropic NDF can be generalized using this matrix [Heitz 2014] as follows:

$$D(\mathbf{h}) = \frac{\chi^+(h_z)}{\pi\sqrt{\det(\mathbf{A})}\left([h_x,h_y]\mathbf{A}^{-1}[h_x,h_y]^{\mathrm{T}}+h_z^2\right)^2}. \tag{17}$$

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

$$\Lambda(\mathbf{o}) = -0.5 + \frac{\sqrt{[o_x,o_y]\mathbf{A}[o_x,o_y]^{\mathrm{T}}+o_z^2}}{2|o_z|}. \tag{18}$$

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 $\mathbf{A}$.

B 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

$$\begin{aligned} m_x &= \cos\psi_y \sin\psi_x, \\ m_y &= \sin\psi_y, \\ m_z &= \cos\psi_y \cos\psi_x. \end{aligned} \tag{19}$$

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 yielded as

$$\begin{aligned} J_{\mathrm{o}\to\perp^m} &= \begin{bmatrix} \frac{\partial m_x}{\partial\psi_x} & \frac{\partial m_x}{\partial\psi_y} \\ \frac{\partial m_y}{\partial\psi_x} & \frac{\partial m_y}{\partial\psi_y} \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi_y\cos\psi_x & -\sin\psi_y\sin\psi_x \\ 0 & \cos\psi_y \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{20}$$

The tangent-space halfvector can be represented using a polar coordinate system as follows:

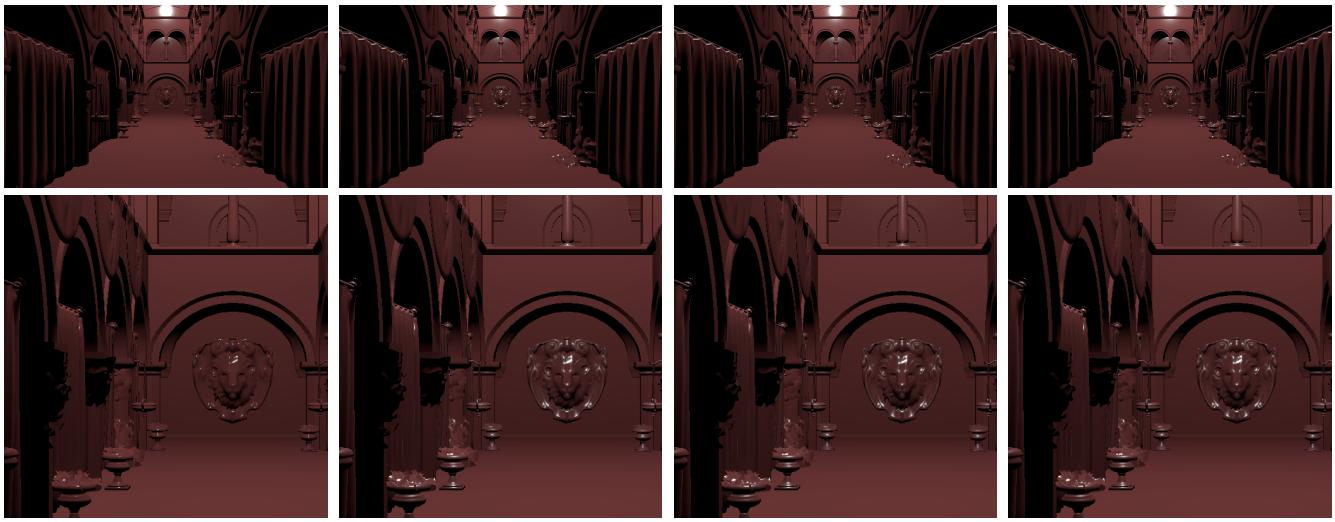
$$\begin{aligned} h_x &= \sin\theta\cos\phi, \\ h_y &= \sin\theta\sin\phi, \\ h_z &= \cos\theta. \end{aligned} \tag{21}$$

Listing 2: *Our roughness calculation for deferred rendering (HLSL).*

```
float3 deltaU = ddx( normal );
float3 deltaV = ddy( normal );
float variance = SCREEN_SPACE_VARIANCE * ( dot( deltaU, deltaU ) + dot( deltaV, deltaV ) );
float kernelSquaredRoughness = min( 2.0 * variance, THRESHOLD );
float squaredRoughness = saturate( roughness * roughness + kernelSquaredRoughness );
```

Listing 3: *Original roughness calculation using the maximum 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 avgNormalTangentSpace = mul( tangentFrame, avgNormal );
float2 avgNormalProjected = avgNormalTangentSpace.xy / abs( avgNormalTangentSpace.z );
float2 deltaU = ddx( avgNormalProjected );
float2 deltaV = ddy( avgNormalProjected );
float2 boundingRectangle = abs( deltaU ) + abs( deltaV );
float maxWidth = max( boundingRectangle.x, boundingRectangle.y );
float variance = SCREEN_SPACE_VARIANCE * maxWidth * maxWidth;
float kernelSquaredRoughness = min( 2.0 * variance, THRESHOLD );
float squaredRoughness = saturate( roughness * roughness + kernelSquaredRoughness );
```



(a) *w/o NDF filtering* **(b)** *Kaplanyan et al. 2016* **(c)** *Ours (Eq.* (13)*)* **(d)** *Maximum eigenvalue (Eq.* (10)*)*

Figure 7: *Isotropic NDF filtering for deferred rendering.*

(a) *Kaplanyan et al. 2016* **(b)** *Ours (Eq.* (13)*)* **(c)** ((**(a)** − **(b)**) × 4. *Red: positive. Green: negative.*

Figure 8: *Visualization of roughness parameter $\bar{\alpha}$ of the filtered isotropic NDF for deferred rendering.*

Using this θ and this ϕ, the rotation from the local-space halfvector to tangent-space halfvector is given by

$$\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & -\sin\phi & \sin\theta\cos\phi \\ \cos\theta\sin\phi & \cos\phi & \sin\theta\sin\phi \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}, \quad (22)$$

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

$$\begin{aligned} J_{\circ\to\perp} &= J_{\perp^m\to\perp} J_{\circ\to\perp^m} \\ &= \begin{bmatrix} \frac{\partial h_x}{\partial m_x} & \frac{\partial h_x}{\partial m_y} \\ \frac{\partial h_y}{\partial m_y} & \frac{\partial h_y}{\partial m_y} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\cos\phi & -\sin\phi \\ \cos\theta\sin\phi & \cos\phi \end{bmatrix} \\ &= \frac{1}{\sqrt{1-h_z^2}} \begin{bmatrix} h_x h_z & -h_y \\ h_y h_z & h_x \end{bmatrix}. \end{aligned} \quad (23)$$

The slope of the halfvector is given as

$$\begin{aligned} h_x^{\parallel} &= -\frac{h_x}{\sqrt{1-h_x^2-h_y^2}}, \\ h_y^{\parallel} &= -\frac{h_y}{\sqrt{1-h_x^2-h_y^2}}. \end{aligned} \quad (24)$$

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

$$J_{\perp\to\parallel} = \begin{bmatrix} \frac{\partial h_x^{\parallel}}{\partial h_x} & \frac{\partial h_x^{\parallel}}{\partial h_y} \\ \frac{\partial h_y^{\parallel}}{\partial h_x} & \frac{\partial h_y^{\parallel}}{\partial h_y} \end{bmatrix} = -\frac{1}{h_z^3} \begin{bmatrix} 1-h_y^2 & h_x h_y \\ h_x h_y & 1-h_x^2 \end{bmatrix}. \quad (25)$$

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

$$J_{\circ\to\parallel} = J_{\perp\to\parallel} J_{\circ\to\perp} = -\frac{1}{h_z^2\sqrt{1-h_z^2}} \begin{bmatrix} h_x & -h_y h_z \\ h_y & h_x h_z \end{bmatrix}. \quad (26)$$

References

BECKMANN, P., AND SPIZZICHINO, A. 1963. Scattering of Electromagnetic Waves from Rough Surfaces. MacMillan.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. ACM Trans. Graph. 1, 1, 7–24.

HEITZ, E. 2014. Understanding the masking-shadowing function in microfacet-based BRDFs. J. Comput. Graph. Tech. 3, 2, 48–107.

KAPLANYAN, A. S., HILL, S., PATNEY, A., AND LEFOHN, A. 2016. Filtering distributions of normals for shading antialiasing. In HPG'16, 151–162.

KAPLANYAN, A. S. 2016. Stable specular highlights. In GDC '16.

SMITH, B. G. 1967. Geometrical shadowing of a random rough surface. IEEE Trans. Antennas and Propagation 15, 5, 668–671.

TROWBRIDGE, T. S., AND REITZ, K. P. 1975. Average irregularity representation of a rough surface for ray reflection. J. Opt. Soc. Am 65, 5, 531–536.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In EGSR'07, 195–206.