Linear Efficient Antialiased Displacement and Reflectance Mapping: Supplemental Material

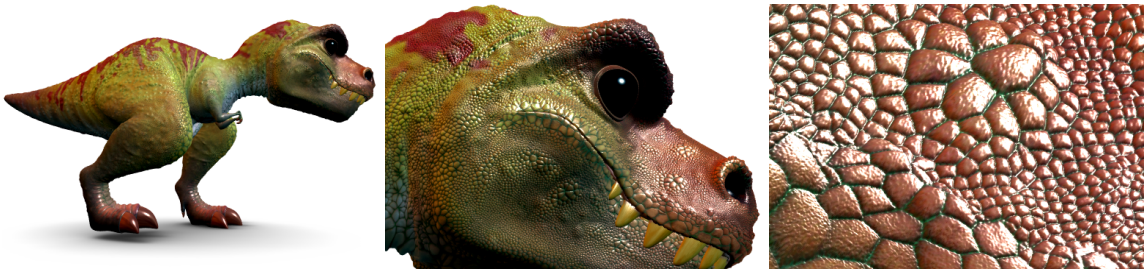
Jonathan Dupuy[1,3] * Eric Heitz[2,1] * Jean-Claude Iehl[3]
Pierre Poulin[1] Fabrice Neyret[2] Victor Ostromoukhov[3]

[1]LIGUM, Dept. I.R.O., Université de Montréal
[2]INRIA/LJK
[3]LIRIS, Université Lyon 1

* Jonathan Dupuy and Eric Heitz are joint first authors

Abstract

This document provides additional material in support of the associated paper. We include in Section 1 the full mathematical development for a number of formulations in the paper. They are not required to understand the theoretical and practical workings of our method, but they may clarify some of the steps taken along these derivations. Section 2 presents several rendered spheres with ground-truth results for normal mapping and displacement mapping, as well as for our method with three sets of sampling grid points. They cover a number of variations of roughnesses, isotropic or anisotropic microfacets distributions, diffuse or specular reflections, and lighting configurations for directional, irradiance maps, or environment maps. The results are discussed with respect to our experience with using LEADR mapping. Finally, a few samples of pseudocodes for the general algorithms and some implementation issues are included.



{jdupuy | jciehl | victor.ostromoukhov}@liris.cnrs.fr
{eric.heitz | fabrice.neyret}@inria.fr
poulin@iro.umontreal.ca

1 Mathematical Derivations

In this section we recall mathematical derivations related to slope distributions and normal distributions used in the paper.

1.1 Jacobian of the Normal-to-Slope Transformation

Here we derive the Jacobian $\left\| \frac{\partial \tilde{n}}{\partial \omega_n} \right\|$ used in Equation (10) in the paper. We note a slope as $\tilde{n} = (x_{\tilde{n}}, y_{\tilde{n}})^t$ and its relation to the normal ω_n is given by

$$\omega_n = \frac{(-x_{\tilde{n}}, -y_{\tilde{n}}, 1)^t}{\sqrt{1 + x_{\tilde{n}}^2 + y_{\tilde{n}}^2}}.$$

In spherical coordinates (θ, ϕ), the normal is given by

$$\omega_n = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^t$$

and the slopes can thus be obtained by

$$\begin{pmatrix} \theta \\ \phi \end{pmatrix} \rightarrow \begin{pmatrix} x_{\tilde{n}} \\ y_{\tilde{n}} \end{pmatrix} = \begin{pmatrix} -\tan\theta\cos\phi \\ -\tan\theta\sin\phi \end{pmatrix}.$$

These coordinates systems are illustrated in Figure 1.

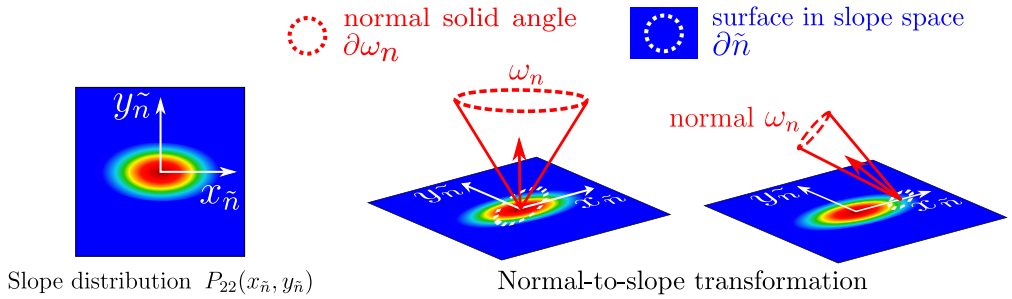


Figure 1: Normal-to-slope transformation. The Jacobian of this transformation is the ratio between the infinitesimal surface element $\partial\tilde{n}$ and $\partial\omega_n$.

The Jacobian of this transformation corresponds to

$$\left\| \frac{\partial \tilde{n}}{\partial(\theta, \phi)} \right\| = \begin{vmatrix} \frac{\partial x_{\tilde{n}}}{\partial\theta} & \frac{\partial x_{\tilde{n}}}{\partial\phi} \\ \frac{\partial y_{\tilde{n}}}{\partial\theta} & \frac{\partial y_{\tilde{n}}}{\partial\phi} \end{vmatrix} = \begin{vmatrix} -\frac{1}{\cos^2\theta}\cos\phi & \tan\theta\sin\phi \\ -\frac{1}{\cos^2\theta}\sin\phi & -\tan\theta\cos\phi \end{vmatrix} = \left| \frac{\sin\theta}{\cos^3\theta} \right|.$$

The Jacobian of the normal-to-slope transformation is

$$\left\| \frac{\partial \tilde{n}}{\partial \omega_n} \right\| = \left\| \frac{\partial \tilde{n}}{\partial(\theta, \phi)} \right\| \left\| \frac{\partial(\theta, \phi)}{\partial \omega_n} \right\| = \left| \frac{\sin\theta}{\cos^3\theta} \right| \frac{1}{|\sin\theta|} = \frac{1}{|\cos^3\theta|}$$

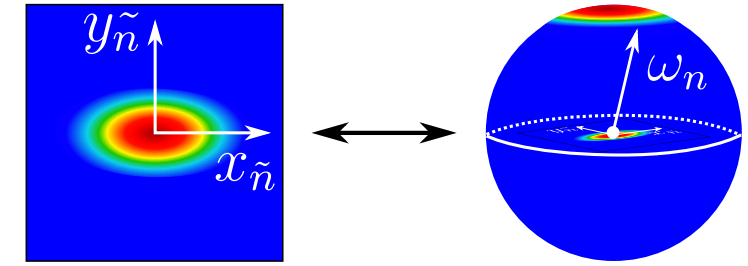
where $\left\| \frac{\partial(\theta,\phi)}{\partial\omega_n} \right\| = \frac{1}{|\sin\theta|}$ is the Jacobian used to transform solid angles into spherical coordinates. Since we define the macronormal $\omega_g = (0, 0, 1)^t$, we have $(\omega_n \cdot \omega_g) = \cos\theta$ and thus

$$\boxed{\left\| \frac{\partial \tilde{n}}{\partial \omega_n} \right\| = \frac{1}{|\omega_n \cdot \omega_g|^3}}. \tag{1}$$

1.2 Beckmann Distribution

The Beckmann distribution is built upon a Gaussian distribution of slopes, as illustrated in Figure 2. The Gaussian slope distribution $P_{22}(\tilde{n})$ is such as

$$1 = \int \int P_{22}(\tilde{n})\, d\tilde{n}.$$



Slope distribution $P_{22}(x_{\tilde{n}}, y_{\tilde{n}})$

Normal distribution $D(\omega_n)$
(Beckmann)

Figure 2: The Beckmann distribution is built upon a Gaussian slope distribution.

By integrating over the normals instead of over the slopes with the variable change $d\tilde{n} = \left\| \frac{\partial \tilde{n}}{\partial \omega_n} \right\| d\omega_n$, we get

$$\begin{aligned} 1 &= \int_\Omega P_{22}(\tilde{n}) \left\| \frac{\partial \tilde{n}}{\partial \omega_n} \right\| d\omega_n \\ &= \int_\Omega \frac{P_{22}(\tilde{n})}{(\omega_n \cdot \omega_g)^3}\, d\omega_n. \end{aligned}$$

In distribution P_{22}, the density of each microfacet slope is weighted by its projected area $(\omega_n \cdot \omega_g)$ over the macrosurface. In the Beckmann distribution D, each normal is represented independently of any reference plane and is weighted by its real area in world space. This is why it comes with the additional factor $\frac{1}{\omega_n \cdot \omega_g}$

$$\begin{aligned} 1 &= \int_\Omega (\omega_n \cdot \omega_g) \frac{P_{22}(\tilde{n})}{(\omega_n \cdot \omega_g)^4}\, d\omega_n \\ &= \int_\Omega (\omega_n \cdot \omega_g)\, D(\omega_n)\, d\omega_n \end{aligned}$$

where the Beckmann distribution is

$$\boxed{D(\omega_n) = \frac{P_{22}(\tilde{n})}{(\omega_n \cdot \omega_g)^4}}. \tag{2}$$

1.3 Jacobian of the Reflection Operation

Here we derive the Jacobian $\left\| \frac{\partial \omega_i}{\partial \omega_n} \right\|$ used in Sections 4.4 and 5.2 of the paper. The reflection operator is defined by $\omega_i = 2\,(\omega_n \cdot \omega_o)\,\omega_n - \omega_o$. To simplify the derivation we chose to center the basis around ω_o instead

of ω_n this time

$$\omega_o = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \omega_n = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}$$

and the reflected vector is then

$$\omega_i = 2 \begin{pmatrix} \cos\phi\sin\theta\cos\theta \\ \sin\phi\sin\theta\cos\theta \\ \cos^2\theta \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

They are illustrated in Figure 3.

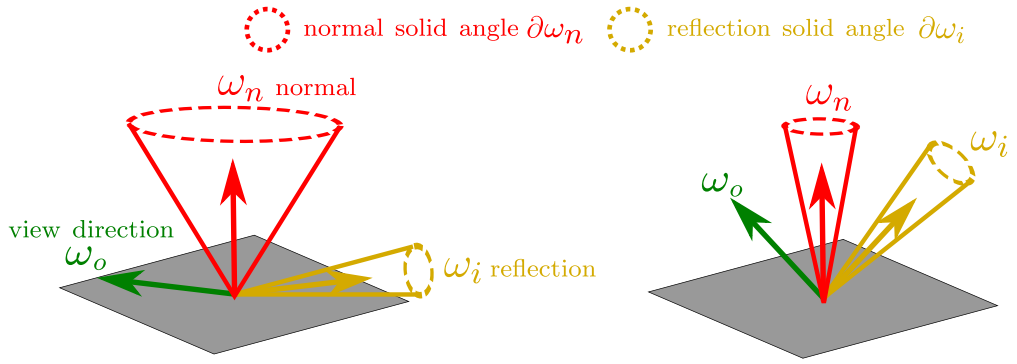


Figure 3: Reflection operation. The Jacobian of this transformation is the ratio between the infinitesimal solid angles $\partial\omega_i$ and $\partial\omega_n$.

The partial derivatives of ω_i are

$$\frac{\partial\omega_i}{\partial\theta} = 2 \begin{pmatrix} \cos\phi\left(\cos^2\theta - \sin^2\theta\right) \\ \sin\phi\sin\theta\cos\theta \\ -2\cos\theta\sin\theta \end{pmatrix} \quad \text{and} \quad \frac{\partial\omega_i}{\partial\phi} = 2 \begin{pmatrix} -\cos\theta\sin\theta\sin\phi \\ \cos\theta\sin\theta\cos\phi \\ 0 \end{pmatrix}$$

and the Jacobian of this transformation is given by the norm of the wedge product

$$\begin{aligned} \left\| \frac{\partial\omega_i}{\partial(\theta,\phi)} \right\| = \left\| \frac{\partial\omega_i}{\partial\theta} \wedge \frac{\partial\omega_i}{\partial\phi} \right\| &= 4 \left\| \begin{vmatrix} \cos\phi\left(\cos^2\theta - \sin^2\theta\right) & \sin\phi\sin\theta\cos\theta & -2\cos\theta\sin\theta \\ -\cos\theta\sin\theta\sin\phi & \cos\theta\sin\theta\cos\phi & 0 \\ \vec{i} & \vec{j} & \vec{k} \end{vmatrix} \right\| \\ &= 4\sqrt{\left(2\cos^2\theta\sin^2\theta\cos\phi\right)^2 + \left(-2\cos^2\theta\sin^2\theta\sin\phi\right)^2 + \left(\cos\theta\sin\theta\left(\cos^2\theta - \sin^2\theta\right)\right)^2} \\ &= 4|\cos\theta\ \sin\theta|. \end{aligned}$$

The solid angle transformation of the reflection operation is the Jacobian

$$\left\| \frac{\partial\omega_i}{\partial\omega_n} \right\| = \left\| \frac{\partial\omega_i}{\partial(\theta,\phi)} \right\| \left\| \frac{\partial(\theta,\phi)}{\partial\omega_n} \right\| = 4\,|\cos\theta\ \sin\theta| \frac{1}{|\sin\theta|} = 4\,|\cos\theta|$$

and since in this basis $(\omega_n \cdot \omega_o) = \cos\theta$, we get

$$\boxed{\left\| \frac{\partial\omega_i}{\partial\omega_n} \right\| = 4\,|\omega_n \cdot \omega_o|}. \tag{3}$$

1.4 The Projected Area

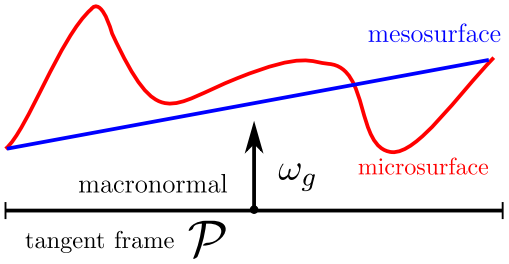
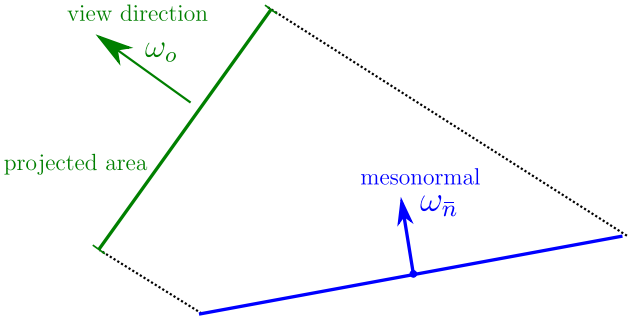
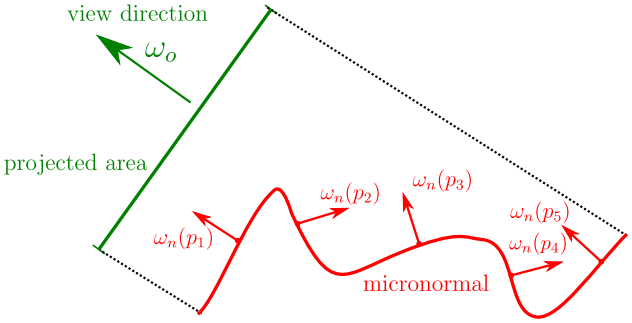
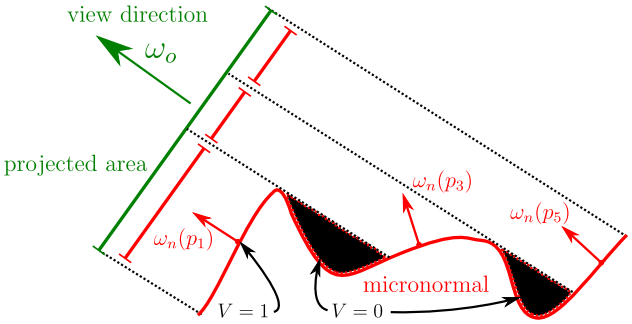
(a) configuration	
(b) projected area $= \dfrac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g}$	
(c) projected area $= \displaystyle\int_{\mathcal{P}} \frac{\omega_n(p) \cdot \omega_o}{\omega_n(p) \cdot \omega_g} k_{\mathcal{P}}(p)\, dp$	
(d) projected area $= \displaystyle\int_{\mathcal{P}} V(p, \omega_o) \frac{\omega_n(p) \cdot \omega_o}{\omega_n(p) \cdot \omega_g} k_{\mathcal{P}}(p)\, dp$	

Table 1: Different computations of the projected area.

The projected area, used in Equation (1) in the paper, is the area covered by the surface in the view

direction, and equals the projected area of the mesonormal as illustrated in Table 1(b)

$$\text{projected area} = \frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g}. \tag{4}$$

Since the normals of the microsurface are associated to the slopes by

$$\frac{\omega_n}{\omega_n \cdot \omega_g} = (-x_{\tilde{n}}, -y_{\tilde{n}}, 1)^t$$

and since the normal of the mesosurface is associated to the average slope by

$$\begin{aligned}\frac{\omega_{\bar{n}}}{\omega_{\bar{n}} \cdot \omega_g} &= (-\mathbb{E}[x_{\tilde{n}}], -\mathbb{E}[y_{\tilde{n}}], 1)^t \\ &= \left(-\int_{\mathcal{P}} x_{\tilde{n}}(p)\, k_{\mathcal{P}}(p)\, dp \ , \ -\int_{\mathcal{P}} y_{\tilde{n}}(p)\, k_{\mathcal{P}}(p)\, dp \ , \ 1 \right)^t\end{aligned}$$

the mesonormal and the micronormals are linked by the relation

$$\begin{aligned}\frac{\omega_{\bar{n}}}{\omega_{\bar{n}} \cdot \omega_g} &= \int_{\mathcal{P}} (-x_{\tilde{n}}(p), -y_{\tilde{n}}(p), 1)^t \ k_{\mathcal{P}}(p)\, dp \\ &= \int_{\mathcal{P}} \frac{\omega_n(p)}{\omega_n(p) \cdot \omega_g}\, k_{\mathcal{P}}(p)\, dp. \end{aligned} \tag{5}$$

By combining Equations (4) and (5), and because of the linearity of the (nonclamped) dot product, we get that the projected area is also the sum of the (nonclamped) projected area of all the microfacets, as illustrated in Table 1(c)

$$\begin{aligned}\text{projected area} &= \frac{\omega_{\bar{n}}}{\omega_{\bar{n}} \cdot \omega_g} \cdot \omega_o \\ &= \left(\int_{\mathcal{P}} \frac{\omega_n(p)}{\omega_n(p) \cdot \omega_g}\, k_{\mathcal{P}}(p)\, dp \right) \cdot \omega_o \\ &= \int_{\mathcal{P}} \frac{\omega_n(p) \cdot \omega_o}{\omega_n(p) \cdot \omega_g}\, k_{\mathcal{P}}(p)\, dp. \end{aligned} \tag{6}$$

The same projected area can be obtained by accounting only for the microfacets that are visible in the projection direction. This implies introducing a Heaviside *masking term* V, that removes microfacets that are not visible ($V = 0$), as illustrated in Table 1(d),

$$\text{projected area} = \int_{\mathcal{P}} V(p, \omega_o)\, \frac{\omega_n(p) \cdot \omega_o}{\omega_n(p) \cdot \omega_g}\, k_{\mathcal{P}}(p)\, dp. \tag{7}$$

1.5 The Masking Term

In this section we show how the masking term is related to the Normal Distribution Function (NDF), and how it is used to normalize the BRDF. Contrary to common belief, we show that *the masking term due to Smith [Smi67] is not an approximation, but is the exact required renormalization coefficient.* We also show how to use this property to avoid some shader computations when the shadowing occurring on the microsurface is not rendered, and thus, does not need to be filtered.

Deriving the Exact Masking Term. From Equations (4) and (7) we get

$$\frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} = \int_{\mathcal{P}} V(p, \omega_o) \, \frac{\omega_n(p) \cdot \omega_o}{\omega_n(p) \cdot \omega_g} \, k_{\mathcal{P}}(p) \, dp.$$

Note that microfacets that are backface culled are always self-masked ($V = 0$), and replacing the dot product by a clamped dot product produces the same result

$$\frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} = \int_{\mathcal{P}} V(p, \omega_o) \, \frac{\langle \omega_n(p), \omega_o \rangle}{\omega_n(p) \cdot \omega_g} \, k_{\mathcal{P}}(p) \, dp$$

where we note $\langle -, - \rangle$ the clamped dot product. Even if it does not immediately change the result, we introduce the clamped dot product because it is required for the next step of this derivation. An important assumption in microfacet theory is that masking $V(p, \omega_o)$ and orientation of *nonbackface culled normals* $\omega_n(p)$ are independent. Integrating only over the nonbackface culled microfacets with the clamped dot product keeps the integrands independent and allows for separate integration

$$\begin{aligned} \frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} &= \int_{\mathcal{P}} V(p, \omega_o) \, k_{\mathcal{P}}(p) \, dp \int_{\mathcal{P}} \frac{\langle \omega_n(p), \omega_o \rangle}{\omega_n(p) \cdot \omega_g} \, k_{\mathcal{P}}(p) \, dp \\ &= \mathbb{E}[V(p, \omega_o)] \; \mathbb{E}[\langle \omega_n(p), \omega_o \rangle] \end{aligned}$$

where $\mathbb{E}[V(p, \omega_o)]$ is the average masking term, and $\mathbb{E}[\langle \omega_n(p), \omega_o \rangle]$ is the average backface culled projected area. By changing the integration domain from surface patch $\mathcal{P}$ to the space of the normals, the average backface culled projected area can be expressed as

$$\begin{aligned} \mathbb{E}[\langle \omega_n(p), \omega_o \rangle] &= \int_{\mathcal{P}} \frac{\langle \omega_n(p), \omega_o \rangle}{\omega_n(p) \cdot \omega_g} \, k_{\mathcal{P}}(p) \, dp \\ &= \int_{\Omega} \langle \omega_n, \omega_o \rangle \, D(\omega_n) \, d\omega_n \end{aligned}$$

where D is the NDF over $\mathcal{P}$.

The average masking over the surface is then entirely determined by the NDF

$$\begin{aligned} \mathbb{E}[V(p, \omega_o)] &= \frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} \, \frac{1}{\mathbb{E}[\langle \omega_n(p), \omega_o \rangle]} \\ &= \frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} \, \frac{1}{\displaystyle\int_{\Omega} \langle \omega_n, \omega_o \rangle \, D(\omega_n) \, d\omega_n}. \end{aligned} \tag{8}$$

If D is a Beckmann distribution, the solution of this equation is exact and expressed as

$$\boxed{\mathbb{E}[V(p, \omega_o)] = \frac{1}{1 + \Lambda(\omega_o)}}$$

where $\frac{1}{1+\Lambda(\omega_o)}$ is the masking/shadowing function from Smith [Smi67]. A derivation of this equation for a noncentered Beckmann distribution is given in the supplemental material of Heitz et al. [HNPN13]. Originally, Smith [Smi67] used another derivation based on a raytracing formulation. Note that the only approximations we made so far are the assumptions of normal/masking independence and of a Gaussian surface (D is a Beckmann distribution).

The Normalization Coefficient. The NDF is such as that the total projected area onto the macrosurface is the surface of patch $\mathcal{P}$ (defined as the value 1 in this document)

$$1 = \int_\Omega (\omega_n \cdot \omega_g)\, D(\omega_n)\, d\omega_n.$$

Similarly, the NDF is also such as that the projected area onto the view direction is the projected area of the mesonormal

$$\begin{aligned}\frac{\omega_{\bar{n}} \cdot \omega_o}{\omega_{\bar{n}} \cdot \omega_g} &= \mathbb{E}[V(p,\omega_o)]\, \mathbb{E}[\langle \omega_n(p), \omega_o \rangle] \\ &= \frac{1}{1+\Lambda(\omega_o)} \int_\Omega \langle \omega_n, \omega_o \rangle\, D(\omega_n)\, d\omega_n. \end{aligned} \tag{9}$$

This shows formally that Smith's masking function $\frac{1}{1+\Lambda(\omega_o)}$ has to be used to ensure the conservation of the projected area, and thus, of the energy of the BRDF. It cannot be replaced with other arbitrarily chosen masking functions.

Misunderstanding of the Masking Term. A common misbelief in the computer graphics community (e.g., [WMLT07, MHM+13]) is that Smith's masking term is an alternative "good approximation" to Cook and Torrance's masking term based on V-cavities [CT82] (usually called "the geometric factor"), and that the choice of the best masking term is arbitrary.

Our derivations show that Smith's formula is not just a good approximation, but is the exact masking term under the assumptions of the model (i.e., Gaussian surface and normal/-masking independence). Smith's masking term provides the exact renormalization coefficient required for the BRDF, is entirely determined by the NDF, and cannot be arbitrarily chosen.

Smith compared his formula to real-world measurements and discovered that it was a good approximation. The approximation does not reside in his derivation because, within the framework of his model, his formula is exact. The approximation resides in the estimation of real-world surfaces with Gaussian statistics, and in the assumption of normal/masking independence.

Note that Smith's formula can be generalized for any kind of NDF D, as shown in Equation (9) (under the assumption of normal/masking independence). Walter et al. [WMLT07] show how to extend Smith's raytracing formulation to derive a generalized form for his formula. It provides the same formula than the solution of our Equation (9). The advantage of our derivations is mainly that it emphasizes the relation with the projected area, and thus that the result is exact rather than approximate.

Using this Knowledge to Simplify Shader Code without Self-Shadowing. Equation (9) can be used to avoid evaluations of the masking term when shadows are not rendered. The shadowing term is $\mathbb{E}[V(p,\omega_o)\, V(p,\omega_i)] = \frac{1}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}$. As explained in Section 6 of the paper, rendering self-shadowing effects over highly detailed displacement maps may be difficult because very high resolution shadow maps need to be used in order to capture the shadows of the small displacements. In real life, one may want to only account for long-range shadows and neglect self-shadowing over the displacement map. In this case, the filtering strategy should not integrate shadowing. Indeed, if shadowing is represented in the filtered BRDF but not computed at close-up views, then transitions between scales may be inconsistent (the filtered appearance with the shadowing term would look darker than the ground truth without shadowing). Therefore, when self-shadowing of the displacement map is not rendered, we replace the masking/shadowing term by the masking term only, by replacing $\frac{1}{1+\Lambda(\omega_o)+\Lambda(\omega_i)}$ by $\frac{1}{1+\Lambda(\omega_o)}$.

If only $\frac{1}{1+\Lambda(\omega_o)}$ has to be evaluated, we can use Equation (9) to avoid a direct evaluation by reusing computations in the shader code. We have

$$\frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} \frac{1}{1+\Lambda(\omega_o)} = \frac{1}{\displaystyle\int_\Omega \langle \omega_n, \omega_o \rangle\, D(\omega_n)\, d\omega_n}.$$

In Algorithms 1 and 3 in the paper, we use sampling schemes where the integration of a quantity over the space of the normals is replaced by a discrete sum over the space of the slopes

$$\int_\Omega (-)\, D(\omega_n)\, d\omega_n \approx \sum_{N^2} (-)\, \frac{W_{\tilde{n}}}{\omega_n \cdot \omega_g},$$

where $W_{\tilde{n}}$ is a precomputed Gaussian weight and $\frac{1}{\omega_n \cdot \omega_g}$ is the projection weighting associated with D, as explained in Section 1.2 of this document. Note that this approximation is exact when $N \to \infty$. Then we can approximate

$$\frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} \frac{1}{1 + \Lambda(\omega_o)} \approx \frac{1}{\sum_{N^2} \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}}. \tag{10}$$

In Algorithms 1 and 3 in the paper, we accumulate the sum $S = \sum_{N^2} \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}$ in the integration loops of the algorithms, thus reusing most of the operations. After the integration loops, we divide the result by S. This spares the multiplication by $\frac{1}{1+\Lambda(\omega_o)} \frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o}$, and thus the evaluation of function Λ.

Here are the simplified versions of Algorithms 1, 2, and 3 from the paper, without self-shadowing, and optimized with this solution. The modifications, compared to the original algorithms in the paper, are indicated in red.

Simplified Algorithm 1. Algorithm 1 presented in the paper numerically integrates

$$I = \frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} \sum_{N^2} \frac{L(\omega_i)\, F(\omega_h, \omega_i)\, \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g}}{1 + \Lambda(\omega_o) + \Lambda(\omega_i)} W_{\tilde{n}}.$$

By dropping the self-shadowing it becomes

$$I = \frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} \sum_{N^2} \frac{L(\omega_i)\, F(\omega_h, \omega_i)\, \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g}}{1 + \Lambda(\omega_o)} W_{\tilde{n}}.$$

and by using Equation (10), it becomes

$$I = \frac{\sum_{N^2} L(\omega_i)\, F(\omega_h, \omega_i)\, \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}}{\sum_{N^2} \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}}.$$

This is what is computed by this simplified version of Algorithm 1. The pseudocode of this algorithm is provided in Section 3 of this document.

Simplified Algorithm 2. Algorithm 2 presented in the paper computes the irradiance map where masking and shadowing are included

$$E(\omega_n) = \int_{\Omega_i} \frac{L(\omega_i)\, \langle \omega_n, \omega_i \rangle}{1 + \Lambda(\omega_o) + \Lambda(\omega_i)}\, d\omega_i.$$

Algorithm 1 Lighting environment map sampling without self-shadowing

function LIGHTINGENVMAPSAMPLING($\omega_o, \mathbb{E}[\tilde{n}], \sigma_x, \sigma_y, c_{xy}$)
  $I = 0$
  $S = 0$
  for $j, k = 1..N$ **do**
    $x_{\tilde{n}} = p_j\,\sigma_x + \mathbb{E}[x_{\tilde{n}}]$
    $y_{\tilde{n}} = (r_{xy}\,p_j + \sqrt{1 - r_{xy}^2}\,p_k)\,\sigma_y + \mathbb{E}[y_{\tilde{n}}]$
    $W_{\tilde{n}} = W_j\,W_k$
    $\omega_n = (-x_{\tilde{n}}, -y_{\tilde{n}}, 1)/\sqrt{x_{\tilde{n}}^2 + y_{\tilde{n}}^2 + 1}$
    $\omega_i = 2\,(\omega_n \cdot \omega_o)\,\omega_n - \omega_o$
    $J = 4 \times |\omega_n \cdot \omega_o| \times |\omega_n \cdot \omega_g|^3$
    $\alpha = J \times A$
    $\text{LOD} = \log_2\left(\sqrt{\frac{\alpha}{4\,\pi}} \times \text{textureSize}\right)$
    $I \mathrel{+}= W_{\tilde{n}} \times F(\omega_n, \omega_i) \times \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} \times \text{textureLod}(\omega_i, \text{LOD})$
    $S \mathrel{+}= W_{\tilde{n}} \times \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g}$
  end for
  return I/S
end function

Without shadowing we just withdraw masking and shadowing (masking will be integrated in "Simplified Algorithm 3")

$$E(\omega_n) = \int_{\Omega_i} L(\omega_i)\,\langle \omega_n, \omega_i \rangle\,d\omega_i.$$

Algorithm 2 Computing the irradiance map without self-shadowing

function INITIRRADIANCEMAPCOEFS
  for $0 \le l \le 2$ and $-l \le m \le l$ **do**
    $L_{lm} = 0$
  end for
  for each directional light of radiance L and direction δ_{ω_i} **do**
    $L_{00} \mathrel{+}= L_{00} + 0.282095 \times L$
    $(L_{11}, L_{10}, L_{1-1}) \mathrel{+}= 0.488603 \times (x_i, z_i, y_i) \times L$
    $(L_{21}, L_{2-1}, L_{2-2}) \mathrel{+}= 1.092548 \times (x_i z_i, y_i z_i, x_i y_i) \times L$
    $L_{20} \mathrel{+}= 0.315392 \times (3 z_i^2 - 1) \times L$
    $L_{22} \mathrel{+}= 0.546274 \times (x_i^2 - y_i^2) \times L$
  end for
  $L_{lm} \mathrel{+}= L_{lm}(\text{environment})$
end function

Simplified Algorithm 3. Algorithm 3 presented in the paper computes

$$I = \frac{\omega_{\tilde{n}} \cdot \omega_g}{\omega_{\tilde{n}} \cdot \omega_o} \frac{1}{\pi} \sum_{N^2} E(\omega_n)\,\frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}$$

where the irradiance map E already contained the masking and shadowing for each light direction. "Simplified Algorithm 2" computes E without masking and shadowing, and thus it needs to be added within

Algorithm 3. We use the same trick as in "Simplified Algorithm 1"

$$I = \frac{1}{\pi} \frac{\sum_{N^2} E(\omega_n) \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}}{\sum_{N^2} \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} W_{\tilde{n}}}.$$

In order to sample a single directional light with diffuse microfacets, it is not necessary to use the irradiance map decomposition done in Algorithm 2. It can directly be done by integrating the irradiance of the single light, thus replacing $E(\omega_n)$ by $L \times \langle \omega_n \cdot \omega_i \rangle$, where ω_i is the direction of the light and L, its radiance.

Algorithm 3 Irradiance map sampling without self-shadowing

function IRRADIANCEMAPSAMPLING($\omega_o, \mathbb{E}[\tilde{n}], \sigma_x, \sigma_y, c_{xy}$)
  $I = 0$
  $S = 0$
  for $j, k = 1..N$ **do**
    $x_{\tilde{n}} = p_j \, \sigma_x + \mathbb{E}[x_{\tilde{n}}]$
    $y_{\tilde{n}} = (r_{xy} \, p_j + \sqrt{1 - r_{xy}^2} \, p_k) \, \sigma_y + \mathbb{E}[y_{\tilde{n}}]$
    $W_{\tilde{n}} = W_j \, W_k$
    $\omega_n = (-x_{\tilde{n}}, -y_{\tilde{n}}, 1) / \sqrt{x_{\tilde{n}}^2 + y_{\tilde{n}}^2 + 1}$
    $I \mathrel{+}= W_{\tilde{n}} \times \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g} \times E(\omega_n)$
    $S \mathrel{+}= W_{\tilde{n}} \times \frac{\langle \omega_n, \omega_o \rangle}{\omega_n \cdot \omega_g}$
  end for
return $I/S \times \frac{1}{\pi}$
end function

2 Shading Models

2.1 Normal Mapping vs. Displacement Mapping

In this section we compare appearances produced by normal mapping and displacement mapping. Both are computed by applying a noise function with controllable Gaussian statistics on a sphere. The roughness statistics are given by slope standard deviations σ_θ and σ_ϕ in local basis $(\omega_\theta, \omega_\phi)$ defined at each point over a sphere, as shown in Figure 4 and Table 2.

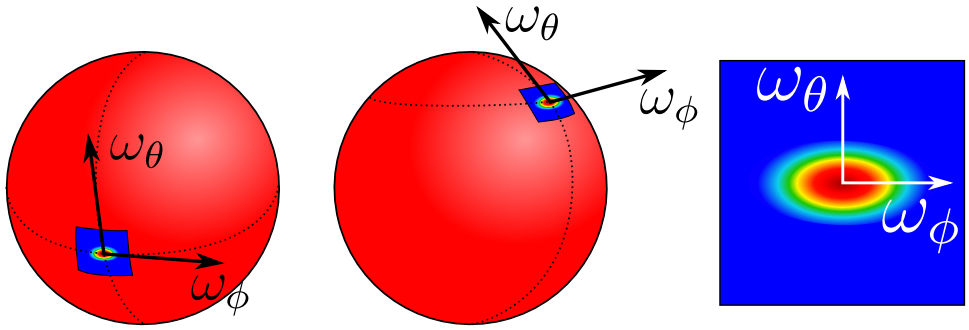


Figure 4: We use a noise primitive that produces Gaussian slope statistics $(\sigma_\theta, \sigma_\phi)$ at any local frame $(\omega_\theta, \omega_\phi)$ on the sphere. The noise statistics are constant in world space and are not affected by the spherical parameterization.

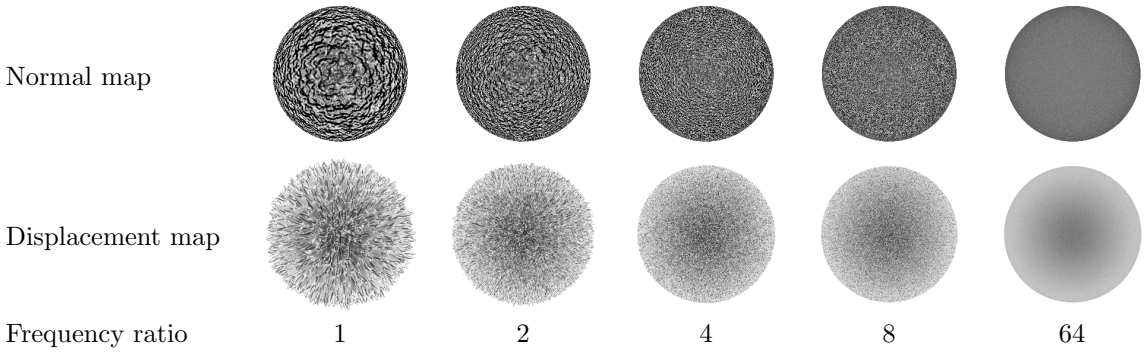
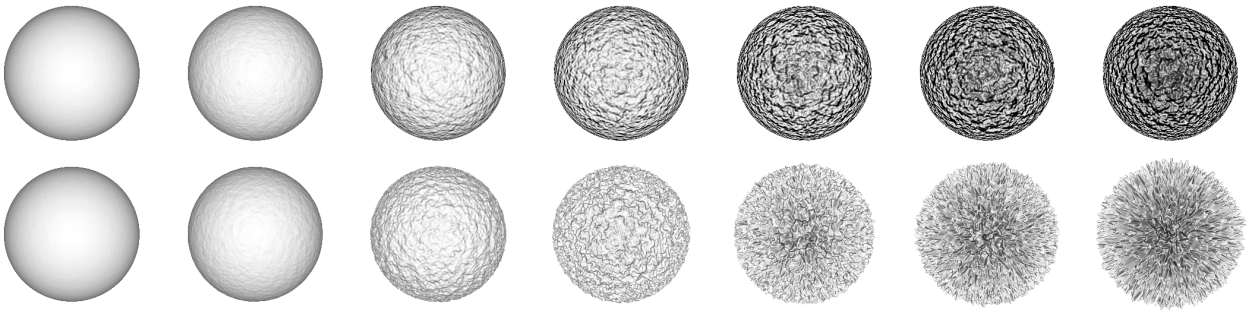


Table 2: We create procedural normals and procedural displacements with a controllable noise producing constant Gaussian statistics over the surface of a sphere. The supersampled shading produced by high-frequency normals or displacements converges toward the BRDF of the noisy surface, and tends to be smooth.

In Table 3 we illustrate the difference at macroscale between normal mapping and displacement mapping. For small roughnesses (e.g., $\sigma < 0.25$), the surface of the sphere is rather smooth, and both mappings produce very similar results. However, as roughness increases, displacements produce more appearance effects due to backface culling and projection weighting. In such situation, faces that point toward the viewer become more visible and the appearances produced by both methods strongly diverge. As a side note, one can also observe the displaced silhouettes resulting in displacement mapping, which do not occur in normal mapping.



$\sigma_\theta = 0.01$	$\sigma_\theta = 0.05$	$\sigma_\theta = 0.25$	$\sigma_\theta = 0.50$	$\sigma_\theta = 1.00$	$\sigma_\theta = 1.50$	$\sigma_\theta = 2.00$
$\sigma_\phi = 0.01$	$\sigma_\phi = 0.05$	$\sigma_\phi = 0.25$	$\sigma_\phi = 0.50$	$\sigma_\phi = 1.00$	$\sigma_\phi = 1.50$	$\sigma_\phi = 2.00$

Table 3: Normal mapping (top row) vs. displacement mapping (middle row) for diffuse microfacets of different roughnesses (bottom row) with one front light, i.e., shining from the camera direction.

2.2 Our Shading Models and Comparisons with Ground Truth

We show the behavior of our BRDF models for different microfacet distributions, lighting configurations, and sampling budgets. We compare results produced at microscale by normal mapping, displacement mapping, and our model. We show results for diffuse microfacets with a single directional light (Tables 4 and 5), with irradiance mapping (Tables 6 and 7), and for specular microfacets with environment lighting (Tables 8 and 9). We show results for isotropic and anisotropic microfacet distributions, where the anisotropy is latitudinal, i.e., with scratches parallel to the equator, or longitudinal, i.e., with scratches running from the North pole to the South pole. In every table handling anisotropy, the different roughnesses (i.e., for σ_θ vs. σ_ϕ) in each row indicates the "strength" of the anisotropy. The low frequency normal and displacement mapped spheres give also a sense of the orientation and "strength" of the anisotropy.

In the following figures, we denote "Normal map" and "Displacement map" the images computed for low- and high-frequencies (subpixel) normal mapping and displacement mapping, respectively, applied in the same way as in Table 3. The results denoted under "LEADR $N \times N$" are the results computed with our BRDF models with a grid of size $N \times N$ for numerical integration.

Finally, in Table 10, we illustrate, for a number of roughness values, the tradeoffs between efficiency and bias of LEADR for specular microfacets, according to the grid size of samples.

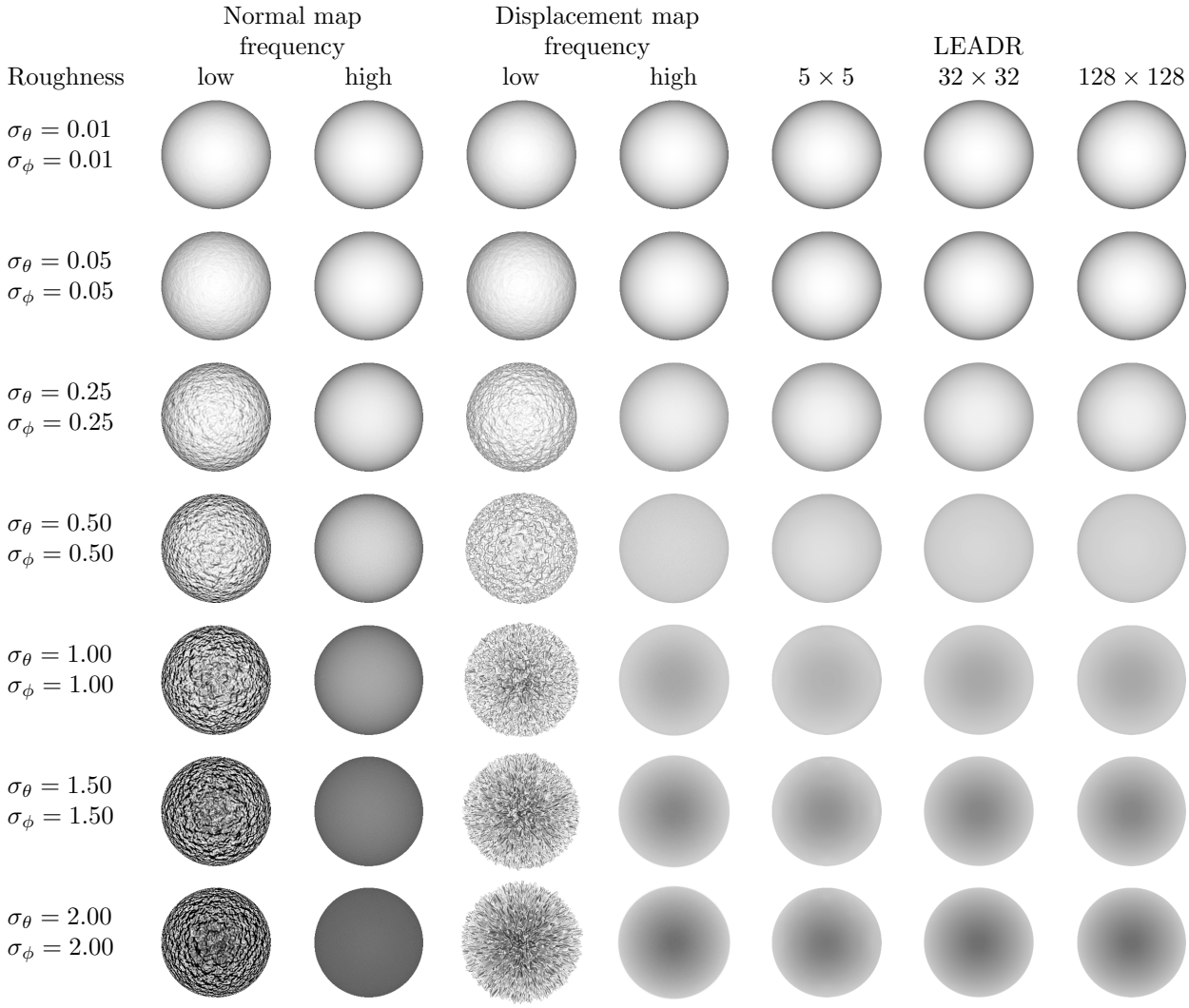


Table 4: Isotropic pure diffuse microfacets and one front light.

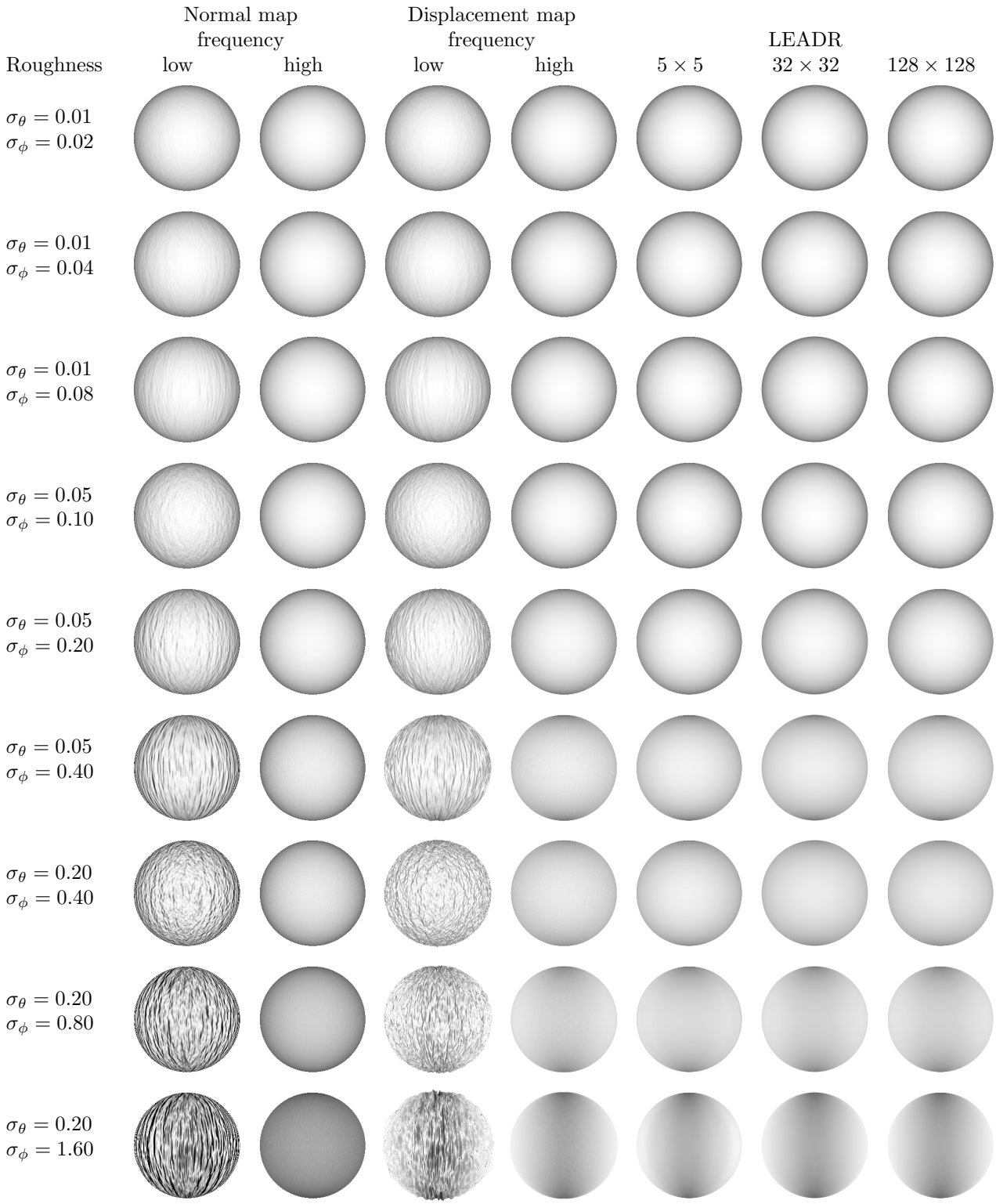


Table 5: Anisotropic pure diffuse microfacets and one front light.

Roughness	Normal map frequency: low	Normal map frequency: high	Displacement map frequency: low	Displacement map frequency: high	LEADR: 5×5	LEADR: 32×32	LEADR: 128×128
$\sigma_\theta = 0.05$ $\sigma_\phi = 0.05$							
$\sigma_\theta = 0.25$ $\sigma_\phi = 0.25$							
$\sigma_\theta = 0.50$ $\sigma_\phi = 0.50$							
$\sigma_\theta = 1.00$ $\sigma_\phi = 1.00$							
$\sigma_\theta = 1.50$ $\sigma_\phi = 1.50$							
$\sigma_\theta = 2.00$ $\sigma_\phi = 2.00$							

Table 6: Isotropic pure diffuse microfacets with an irradiance map.

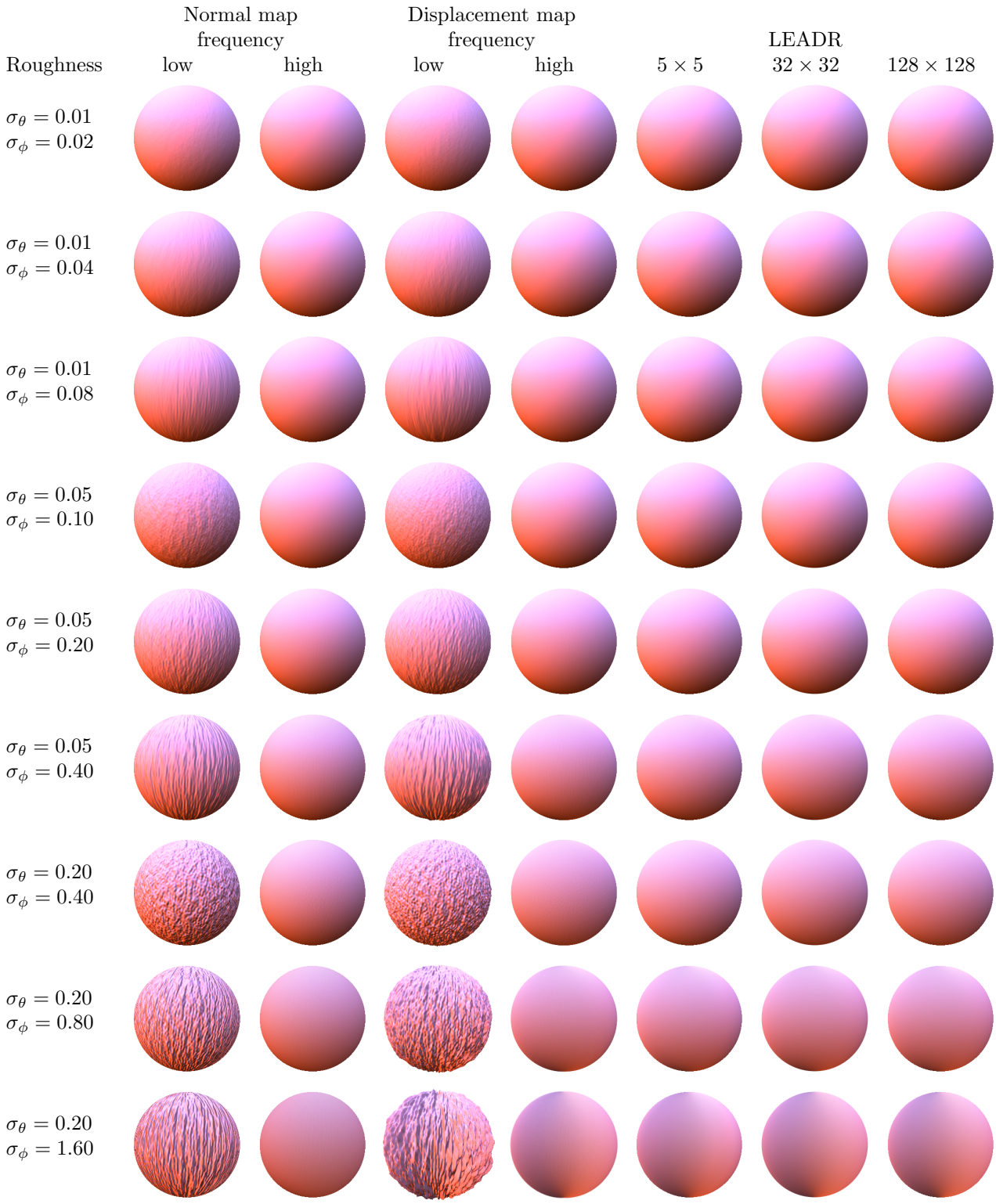


Table 7: Anisotropic pure diffuse microfacets with an irradiance map.

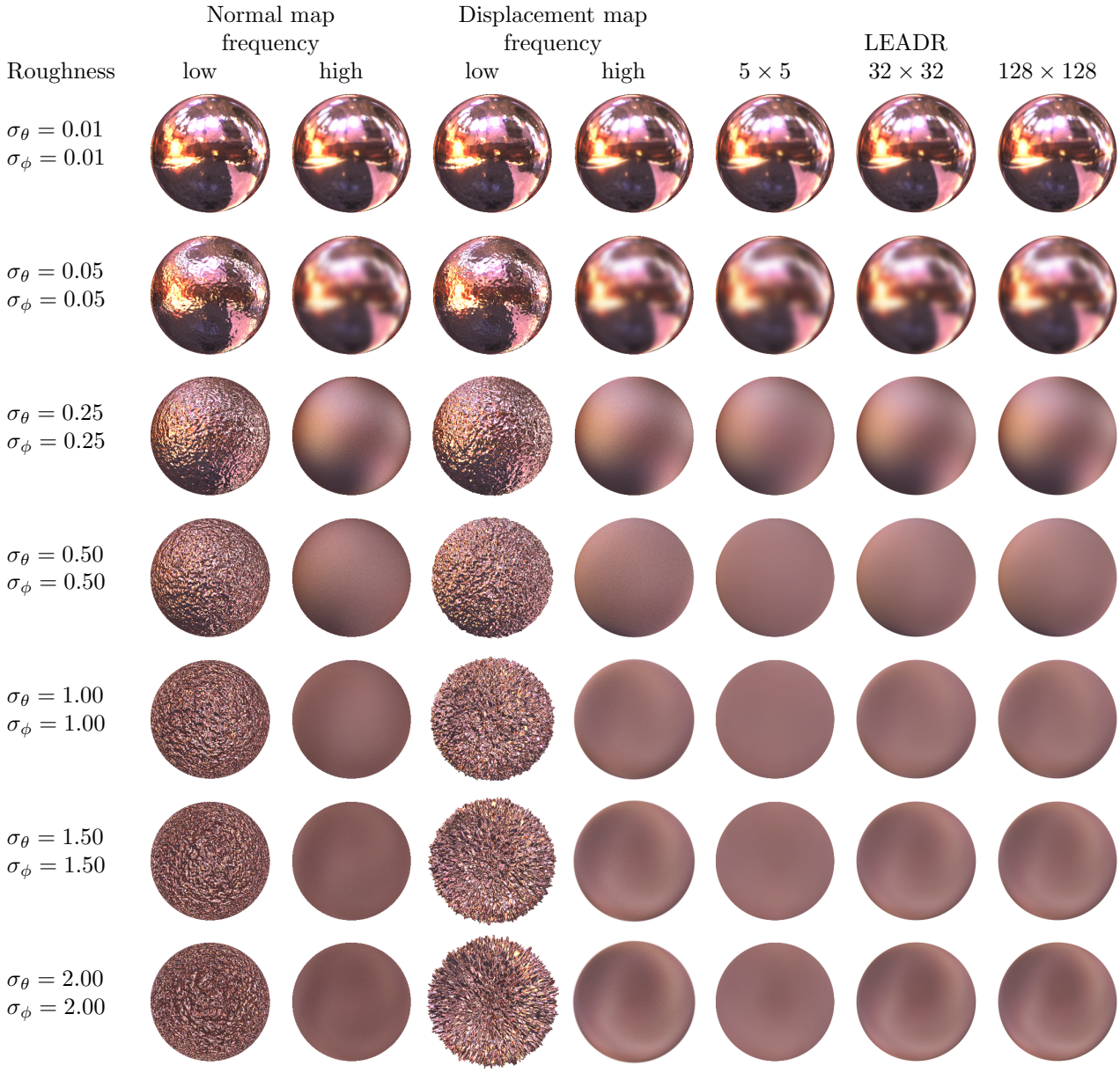


Table 8: Isotropic pure specular microfacets with an environment lighting.

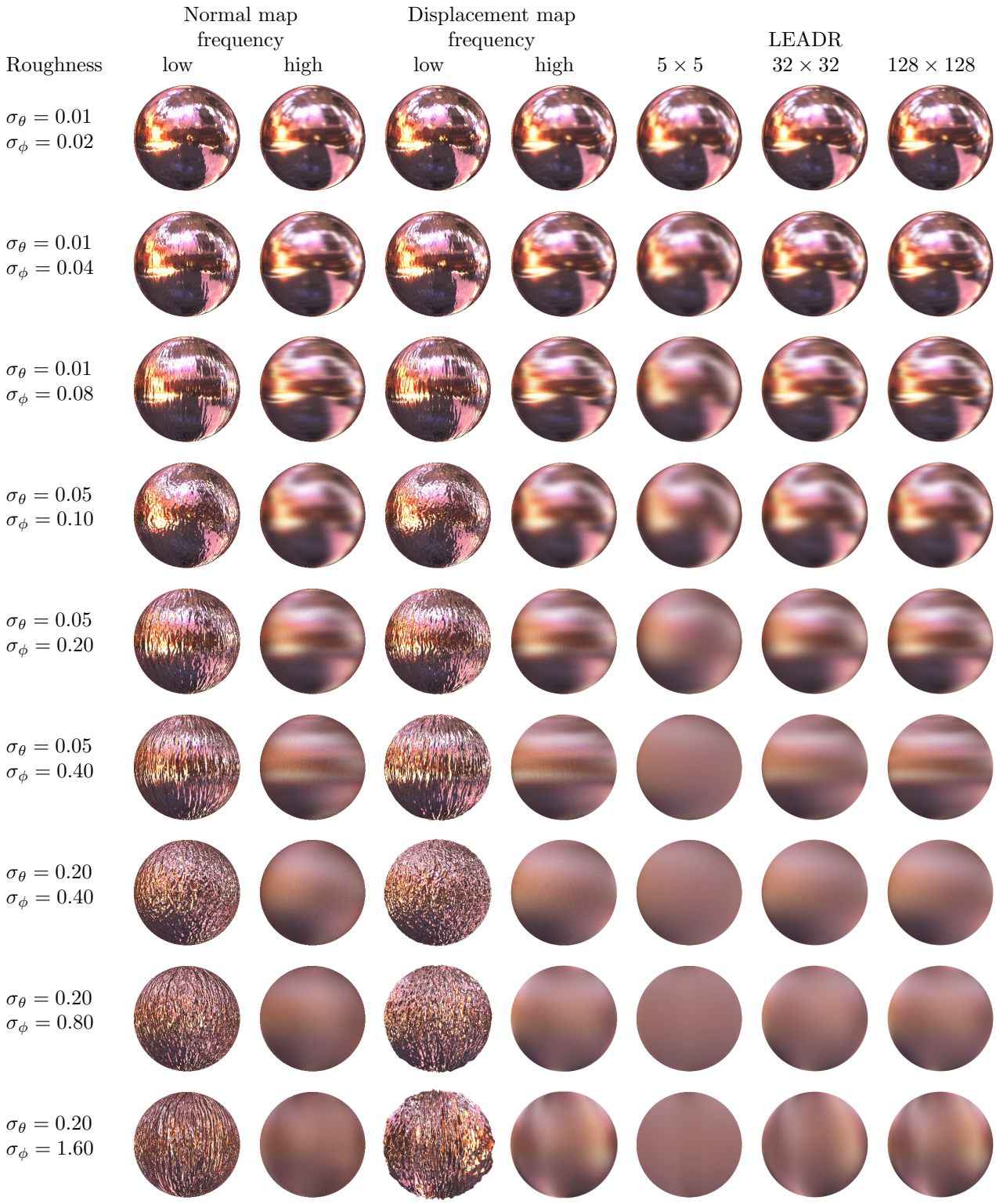


Table 9: Anisotropic pure specular microfacets with an environment lighting.

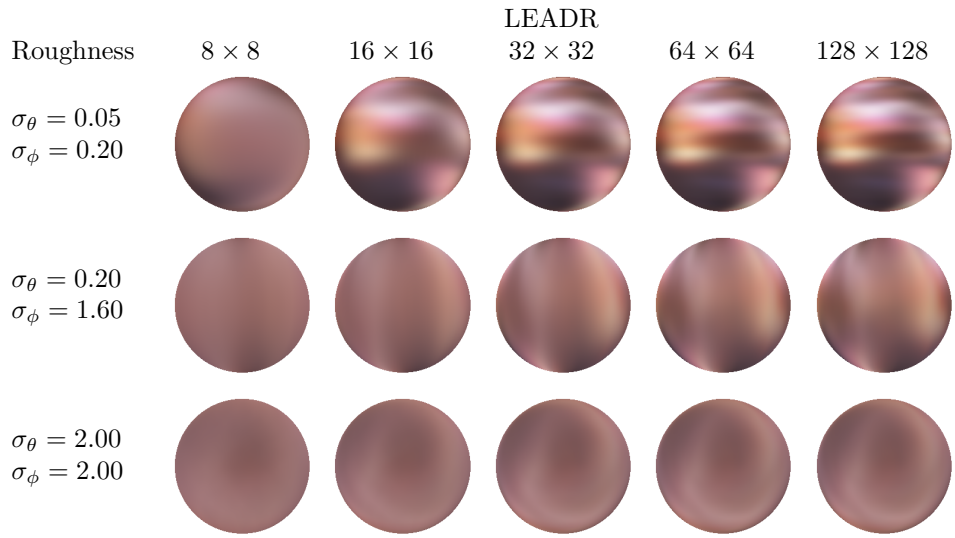


Table 10: Convergence of LEADR environment map sampling scheme. The tradeoff between efficiency and bias can be adjusted by choosing the number of samples.

2.3 Discussion

We can draw a number of conclusions from our experience with our shading models, as discussed below.

These results emphasize the fact that normal mapping and displacement mapping diverge, and do not produce the same appearance when the roughness is important. This shows that even the most sophisticated normal mapping filtering methods, such as [HSRG07], will fail at capturing the appearance of displacement maps since the best they can achieve is the ground-truth appearance of normal maps. On the other hand, our filtering model integrates backface culling and projection weighting specific to displacement maps. When the number of samples N is important, the appearance produced by our BRDF model converges toward the appearance produced by high-frequency displacement maps with the same Gaussian roughness parameters. This validates that our model is exact when applied to Gaussian surfaces.

Concerning the choice of the number of samples, first, for diffuse microfacets, a sampling of 5×5 produces all the important visual features, and there is no differences between a sampling of 9×9 and the converged result computed with a 128×128 grid. To get the best performance/quality tradeoff, we recommend to use 5×5 samples for diffuse microfacets.

Second, for specular microfacets, Figure 5 shows continuous reconstruction filters used to sample an environment map. We compare the filter reconstructed by sampling distribution P_{22} with our grid approach and with importance sampling. For small values of N, importance sampling is too sensitive to the random samples configuration; the uniform grid provides better results.

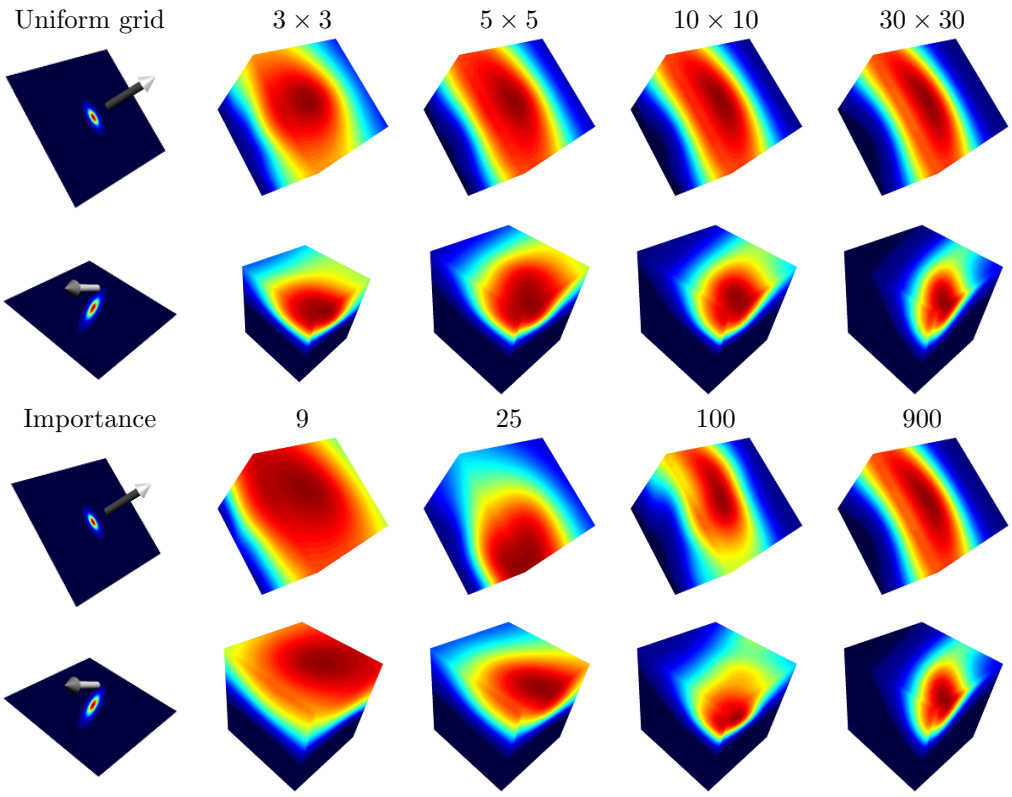


Figure 5: Continuous reconstruction filter for sampling with (top) a uniform grid sampling and (bottom) with importance for two view directions (arrow in each left image). In the four columns on the right, reconstruction from the sampling in the associated cubemap.

Third, our sampling scheme for specular microfacets with environment lighting ensures noise-free results. However, it requires an important number of samples to converge for stronger roughnesses (e.g., $\sigma > 0.25$) as

shown in Table 10. This is particularly true for anisotropic microfacet distributions. The tradeoff between efficiency and bias can be adjusted by choosing the number of samples.

3 Implementation

In this section, we provide commented pseudocode for the fragment shaders associated with our methods. We also discuss some implementation issues.

3.1 Distributions

The 2D Gaussian slope distribution is defined by

$$P_{22}(x_{\tilde{n}}, y_{\tilde{n}}) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\begin{pmatrix} x_{\tilde{n}} - \mathbb{E}[x_{\tilde{n}}] \\ y_{\tilde{n}} - \mathbb{E}[y_{\tilde{n}}] \end{pmatrix}^t \Sigma^{-1} \begin{pmatrix} x_{\tilde{n}} - \mathbb{E}[x_{\tilde{n}}] \\ y_{\tilde{n}} - \mathbb{E}[y_{\tilde{n}}] \end{pmatrix}\right).$$

```
float
P22 (
    float x, float y,
    float x0, float y0, // average slope
    float var_x, float var_y, float c_xy // slope covariance matrix
)
{
        x -= x0;
        y -= y0;
        float det = var_x*var_y - c_xy*c_xy;
        float arg_exp = -0.5 * (x*x*var_y + y*y*var_x - 2.0*x*y*c_xy) / det;
        float P22_ = 0.15915 * inversesqrt(det) * exp(arg_exp);
        return P22_;
}
```

The Beckmann distribution is defined by

$$D(\omega_n) = \frac{P_{22}(\tilde{n})}{(\omega_n \cdot \omega_g)^4}.$$

```
float
D (
    vec3 N,
    float x0, float y0, // average slope
    float var_x, float var_y, float c_xy // slope covariance matrix
)
{
        x = -N.x / N.z; // slope x
        y = -N.y / N.z; // slope y
        float P22_ = P22(x, y, x0, y0, var_x, var_y, c_xy);
        float D_ = P22_ / (N.z * N.z * N.z * N.z);
        return D_;
}
```

3.2 Function Λ

Function Λ is used in the masking and shadowing terms. For a view/light direction defined by $\omega = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$, the Gaussian slope distribution projected along this direction is defined by its average and variance

$$\mu(\phi) = \cos\phi\,\mathbb{E}[x_{\tilde{n}}] + \sin\phi\,\mathbb{E}[y_{\tilde{n}}]$$
$$\sigma^2(\phi) = \cos^2\phi\,\sigma_x^2 + \sin^2\phi\,\sigma_y^2 + 2\,\cos\phi\,\sin\phi\,c_{xy}.$$

We define $\nu = \frac{\cot\theta - \mu(\phi)}{\sigma(\phi)\sqrt{2}}$ and use it with the approximation from Walter et al. [WMLT07]:

$$\Lambda(\omega) \;\approx\; \begin{cases} \frac{1.0 - 1.259\nu + 0.396\nu^2}{3.535\nu + 2.181\nu^2} & \text{if } \nu < 1.6 \\ 0 & \text{otherwise.} \end{cases}$$

```
float
Lambda (
    vec3 V,                 // view vector (world space)
    vec3 X, vec3 Y, vec3 Z, // local basis (world space)
    float x0, float y0,     // average slope
    float var_x, float var_y, float c_xy // slope covariance matrix
)
{
        // view vector projected in plane (X, Y)
        vec2 V_xy = normalize(vec2(dot(V, X), dot(V, Y)));

        // slope statistics in direction V_xy
        float mu_phi = V_xy.x * x0 + V_xy.y * y0;
        float var_phi = V_xy.x*V_xy.x*var_x + V_xy.y*V_xy.y*var_y
                        + 2.0*V_xy.x*V_xy.y*c_xy;

        // slope of view vector
        float cos_theta_v = dot(V, Z);
        float mu_v = cos_theta_v / sqrt(1.0 - cos_theta_v*cos_theta_v);

        // Walter's approximation
        float nu = clamp( (mu_v-mu_phi)*inversesqrt(2.0*var_phi), 0.001, 1.599);
        float Lambda_ = (1.0-1.259*nu_v+0.396*nu_v*nu_v) / (3.535*nu_v+2.181*nu_v*nu_v);
        return Lambda_;
}
```

3.3 Analytical Evaluation of a Rough Specular BRDF with a Point Light Source

For a point/directional lighting from direction ω_i with radiance L we derived the analytical form

$$I = \frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} \frac{L\, F(\omega_h, \omega_i)}{4\,[1 + \Lambda(\omega_o) + \Lambda(\omega_i)]}\, D(\omega_h).$$

Note that since $\omega_g = (0, 0, 1)^t$ the inverse projected area can be written

$$\frac{\omega_{\bar{n}} \cdot \omega_g}{\omega_{\bar{n}} \cdot \omega_o} = \frac{1}{(-\mathbb{E}[x_{\tilde{n}}], -\mathbb{E}[y_{\tilde{n}}], 1)^t \cdot \omega_o}.$$

```
float
roughSpecularPointLight (
    vec3 V, vec3 L,          // view and light vectors (world space)
    vec3 X, vec3 Y, vec3 Z, // local basis (world space)
    float x0, float y0,     // average slope
    float var_x, float var_y, float c_xy // slope covariance matrix
)
{
        // half vector
        vec3 H = normalize(V+L);
        vec3 Hxyz = normalize(vec3(dot(H, X), dot(H, Y), dot(H, Z)));
        if(Hxyz.z <= 0.0) return 0.0;

        // evaluate Beckmann distribution D at half vector
        float d = D(H, x0, y0, var_x, var_y, c_xy);

        // masking and shadowing
        float Lambda_V = Lambda(V, X, Y, Z, x0, y0, var_x, var_y, c_xy);
        float Lambda_L = Lambda(L, X, Y, Z, x0, y0, var_x, var_y, c_xy);
        float MaskingAndShadowing = 1.0 / (1.0 + Lambda_V + Lambda_L);

        // inverse projected area
        float inverse_projected_area = 1.0 / dot(-x0*X-y0*Y+Z), V);

        // radiance
        float I = 0.25 * inverse_projected_area * L * d * MaskingAndShadowing;
        return I;
}
```

Note that in this implementation:

- We dropped the Fresnel term $F(\omega_h, \omega_i)$. Since it is just a separable scaling factor, it can be easily added.
- We account for masking and shadowing. If self-shadowing over the displaced surface is not accounted for, one would include only masking and drop the term $\Lambda(\omega_i)$.

3.4 Integration Samples

We propose an example of a 5×5 sampling grid. We empirically observed that a spacing of 0.9 between the samples delivers the best results for this grid size. For larger values of N we increase the spacing value, though it is not necessary to place samples beyond $[-3.0, 3.0]$, where the normal distribution evaluates to almost 0.

```
const int nbSamples = 5;
const float p[5] = float[5](-1.8, -0.9, 0.0, 0.9, 1.8);
const float w[5] = float[5](0.0725, 0.2443, 0.3663, 0.2443, 0.0725);
const float sampleLength = 0.9; // length of a sample in slope space
```

3.5 Numerical Integration of a Rough Specular BRDF with a Cube Map

This is an implementation of "Simplified Algorithm 1" presented in this document. It does not account for self-shadowing over the displacement map.

```
vec3
roughSpecularCubeMap (
      vec3 V, // view vector (world space)
      vec3 X, vec3 Y, vec3 Z, // local basis (world space)
      float x0, float y0, // average slopes
      float var_x, float var_y, float c_xy // covariance matrix
)
{
        float sigma_x = sqrt(var_x);
        float sigma_y = sqrt(var_y);
        float r_xy = c_xy / (sigma_x*sigma_y);
        float r_xy_ = sqrt(1.0 - r_xy*r_xy);
        float LODoffset =
                -1.48+log2(sampleLength*max(sigma_x, sigma_y)*float(textureSize(cube_map, 0).x));

        // numerical integration
        float S = 0.0;
        vec3 I = vec3(0.0);
        for(int i=0 ; i<nbSamples ; ++i)
        for(int j=0 ; j<nbSamples ; ++j)
        {
                // microfacet slope
                float x = x0 + p[i]*sigma_x;
                float y = y0 + (p[i]*r_xy + p[j]*r_xy_)*sigma_y;
                // microfacet normal
                vec3 N = normalize(- x*X - y*Y + Z);
                // microfacet projected area
                float dotVN = max(0.0, dot(V, N));
                float dotZN = dot(Z, N);
                float S_ = dotVN / dotZN;
                // reflected vector
                vec3 R = reflect(-V, N);
                // Jacobian : slope space area to solid angle
                float J = abs(4.0 * dotZN*dotZN*dotZN * dotVN);
                // LOD in cube map
                float LOD = 0.72*log(J) + LODoffset;
                // radiance
                vec3 I_ = textureLod(cube_map, R, LOD).rgb;
                // sum up
                S += w[i]*w[j] * S_;
                I += w[i]*w[j] * S_ * I_;
        }
        return I/S;
}
```

The computations linked to the LOD are chosen in the following way: the solid angle of a sample is $\alpha = J \times \text{sampleLength}^2 \times \max(\sigma_x, \sigma_y)^2$ and the associated angle is

$$\begin{aligned}\theta &\approx \sqrt{\alpha/\pi} \\ &= \text{sampleLength} \times \max(\sigma_x, \sigma_y) \times \sqrt{J/\pi}.\end{aligned}$$

We approximate the angle of a cube map texel by $\pi/(2 \times \text{TextureSize})$, and we compute the LOD by

$$\begin{aligned}\text{LOD} &= \log_2(2\,\theta \times \text{TextureSize}/\pi) \\ &= \log_2(2\,\pi^{-3/2} \times \text{sampleLength} \times \text{TextureSize} \times \max(\sigma_x, \sigma_y) \times \sqrt{J}) \\ &\approx -1.48 + \log_2(\text{sampleLength} \times \text{TextureSize} \times \max(\sigma_x, \sigma_y)) + 0.72\,\log(J).\end{aligned}$$

Note that when computing the solid angle of the reflection by multiplying the sample area in slope space by the Jacobian of this transformation, we make the assumption that this Jacobian is constant within the surface covered by the sample. In practice, when the number of samples is small, this assumption is not true and we have to slightly bias the level of detail to avoid banding artifacts. For instance, in an implementation with 5×5 samples, we replace the constant -1.48 by 0.5, with 32×32 samples we replace -1.48 by -0.5, and with 128×128, it does not need to be overestimated.

3.6 Numerical Integration of a Rough Diffuse BRDF with an Irradiance Map

This is an implementation of "Simplified Algorithm 3" presented in this document. It does not account for self-shadowing over the displacement map.

```
vec3
roughDiffuse (
      vec3 V, // view vector (world space)
      vec3 X, vec3 Y, vec3 Z, // local basis (world space)
      float x0, float y0,     // average slopes
      float var_x, float var_y, float c_xy // covariance matrix
)
{
        float sigma_x = sqrt(var_x);
        float sigma_y = sqrt(var_y);
        float r_xy = c_xy / (sigma_x*sigma_y);
        float r_xy_ = sqrt(1.0 - r_xy*r_xy);

        // numerical integration
        float S = 0.0;
        vec3 I = vec3(0.0);
        for(int i=0 ; i<nbSamples ; ++i)
        for(int j=0 ; j<nbSamples ; ++j)
        {
                // microfacet slope
                float x = x0 + p[i]*sigma_x;
                float y = y0 + (p[i]*r_xy + p[j]*r_xy_)*sigma_y;
                // microfacet normal
                vec3 N = normalize(- x*X - y*Y + Z);
                // microfacet projected area
                float dotVN = max(0.0, dot(V, N));
                float dotZN = dot(Z, N);
                float S_ = dotVN / dotZN;
                // irradiance
                vec3 I_ = irradiance(N);
                // sum up
                S += w[i]*w[j] * S_;
                I += w[i]*w[j] * S_ * I_;
        }
        return I/S;
}
```

Note that:

- The function "irradiance" is the function whose coefficients are computed in "Simplified Algorithm 2".
- If one wants to render a single directional or point light source of radiance L from direction ω_i, the call to function "irradiance" can be replaced by $L \times \langle \omega_n, \omega_i \rangle$.

```
vec3 I_ = vec3(max(0.0, dot(N, L)));
```

3.7 Problems with a Low Tessellation Rate

As shown in Table 1 of this document, the projected area is an important concept in the definition of LEADR mapping. Indeed, the result of LEADR mapping is undefined when the projected area of the mesonormal is negative. This is not a problem in theory because backface culling should correctly handle negative mesonormals, thus avoiding to render these surface elements. However, in practice, to avoid geometrical aliasing, the tessellation rate ($\approx$ 1 triangle/4 pixels) is lower than the sampling rate of the LEADR texture maps. This sampling rate difference between the triangles and the textures produces a small divergence between the triangle geometric normal and the mesonormal from the LEADR map. Mesonormals that should be backface culled in theory may therefore be rendered, and this can result in undefined values returned from our sampling algorithms. This typically occurs along the silhouettes, where surface normals are almost orthogonal to the view direction, as shown in Figure 6.

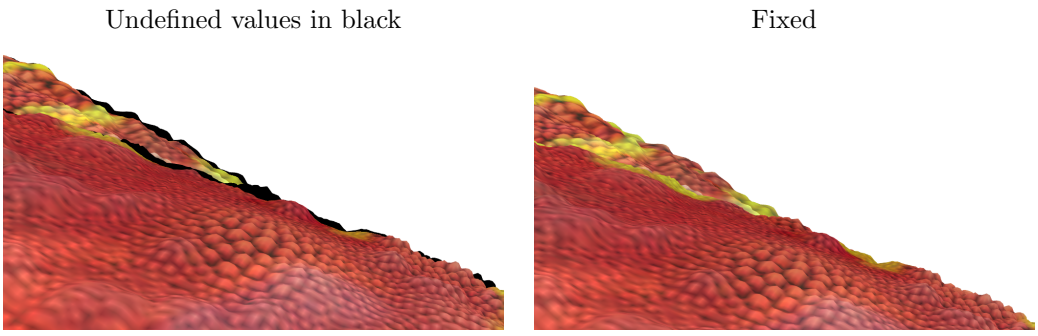


Figure 6: (Left) Mesonormals that should be backface culled may be rendered at the silhouettes when the tessellation rate is lower than the texture sampling rate. In such situations, our algorithms may return undefined values (black pixels). (Right) We solve this problem by adding an artificial micronormal to our sampling schemes in order to ensure that the output of our algorithms is always well defined.

We solve this problem in our sampling schemes by adding an artificial micronormal $\tilde{\omega}_n$. We compute the mesonormal and biased it toward the view direction, thus ensuring that it produces a positive projected area

$$\tilde{\omega}_n = \frac{\omega_{\bar{n}} - (\omega_{\bar{n}} \cdot \omega_o)\,\epsilon\,\omega_o}{||\omega_{\bar{n}} - (\omega_{\bar{n}} \cdot \omega_o)\,\epsilon\,\omega_o||},$$

where we set $\epsilon = 1.01$ in practice. The construction of an artificial micronormal $\tilde{\omega}_n$ is illustrated in Figure 7. We use $\tilde{\omega}_n$ in our sampling algorithms in the same way as the other micronormals, and associate to it a very small weight $\tilde{W}_{\tilde{n}}$ (we set $\tilde{W}_{\tilde{n}} = 0.00001$ in practice).

When the projected area of the other micronormals is positive (this should always be the case in theory), the contribution of $\tilde{\omega}_n$ is negligible because of its small weight $\tilde{W}_{\tilde{n}}$. When the projected area of all the other normals is 0 (this should never happen in theory), only $\tilde{\omega}_n$ contributes to the final shading value. The transition between these two cases is continuous, and the computation of $\tilde{\omega}_n$ makes this "trick" temporally coherent and seamless.

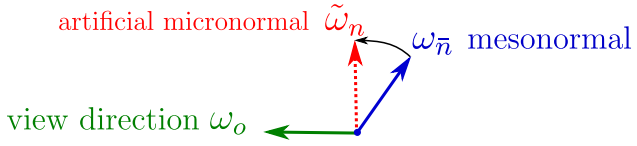


Figure 7: Artificial visible micronormal construction.

References

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.

[HNPN13] E. Heitz, D. Nowrouzezahrai, P. Poulin, and F. Neyret. Filtering color mapped textures and surfaces. In *Proc. Symp. on Interactive 3D Graphics and Games*, pages 129–136. ACM, 2013.

[HSRG07] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Trans. on Graphics*, 26(3):28:1–11, July 2007.

[MHM+13] S. McAuley, S. Hill, A. Martinez, R. Villemin, M. Pettineo, D. Lazarov, D. Neubelt, B. Karis, C. Hery, Naty Hoffman, and H. Zap Andersson. Physically based shading in theory and practice. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, pages 22:1–8. ACM, 2013.

[Smi67] B. Smith. Geometrical shadowing of a random rough surface. *IEEE Trans. on Antennas and Propagation*, 15:668–671, 1967.

[WMLT07] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. Eurographics Symposium on Rendering*, EGSR'07, pages 195–206, 2007.