Rendering Layered Materials with Anisotropic Interfaces

Philippe Weier
Unity Technologies, EPFL

Laurent Belcour
Unity Technologies

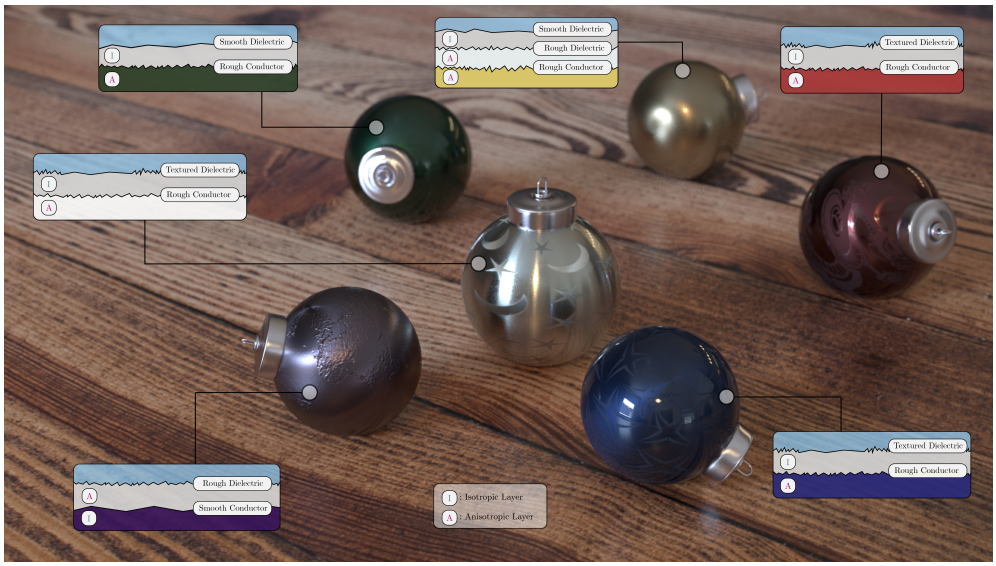


Figure 1. Our model enables the rendering of layered BSDFs with anisotropic interfaces.

Abstract

We present a lightweight and efficient method to render layered materials with anisotropic interfaces. Our work extends the statistical framework of Belcour [2018] to handle anisotropic microfacet models. A key insight to our work is that when projected on the tangent plane, BRDF lobes from an anisotropic GGX distribution are well approximated by ellipsoidal distributions aligned with the tangent frame: its covariance matrix is diagonal in this space. We leverage this property and perform the adding-doubling algorithm on each anisotropy axis independently. We further update the mapping of roughness to directional variance and the evaluation of the average reflectance to account for anisotropy. We extensively tested this model against ground truth.

1. Introduction

In physically-based rendering engines, appearance models are key to produce realistic images [Burley 2012]. One way to enrich the set of materials models is to incorporate structure at the surface of objects, such as a brushing direction for metals. This is typically done with anisotropic material models. Another way to enrich the models is to build more complex material models by simulating light bouncing between many strata before leaving the surface. This can be achieved using layered materials models [Jakob et al. 2014; Belcour 2018; Zeltner and Jakob 2018; Guo et al. 2018] where the surface of an object consists of many plane-parallel interfaces with microfacet geometries.

In this work, we aim to build a new reflectance model for anisotropic layered materials that is compatible with real-time constraints. Since there is no closed form to evaluate anisotropic layered materials, numerical methods are required. For efficient evaluation, one can precompute the layered bidirectional reflectance distribution function (BRDF) and store it using an harmonic representation [Jakob et al. 2014; Zeltner and Jakob 2018]. Another way to robustly evaluate the layered BRDF is to stochastically evaluate it [Guo et al. 2018]. However, these methods require either long per-material precomputations, large storage requirements, or high sample counts to be correctly evaluated. We build our model on a more efficient and lightweight method that approximates light transport in layered BRDFs and achieves real-time performance [Belcour 2018].

In this paper, we extend this latter model to render anisotropic layered materials in an offline scenario. Our key insight is that BRDF lobes from anisotropic GGX normal distribution, once projected on the tangent plane, are well approximated by an ellipsoidal distribution aligned with the tangent-frame's axis. We leverage this property to extend the isotropic model to handle anisotropy.

Concurrently, Yamaguchi et al. [2019] extended the layered BRDF model of Belcour to handle interfaces with anisotropic normal distributions. Both our work and theirs are built on the observation that projected anisotropic BRDF lobes are aligned with the tangent frame. They further looked at the case where the surface's anisotropy is not aligned with the tangent frame, which we omit here; their analysis would apply as well to our method. Because real-time pre-integration of light sources with anisotropic BRDFs is not yet feasible, they targeted interactive performance and relied on Monte-Carlo integration. Here, we focus on rendering quality rather than on efficiency. Also, our work goes a step further and extends two elements that Yamaguchi et al. reused from the isotropic model: the mapping of roughness to variance and the evaluation of the average reflectance at an interface.

Our contributions are the following:

- An analysis of how anisotropic BRDF lobes behave differently in comparison to isotropic ones in the statistical framework of Belcour and how we can generalize the isotropic model using a decorrelation approximation (Section 3.1).

- A new method to efficiently evaluate the average reflectance of an anisotropic rough surface based on Schlick's Fresnel approximation (Section 3.2).
- A new method to find a more accurate conversion of roughness to variance, how it can be used to represent anisotropic materials, and the improvements it brings to the isotropic case (Section 3.3).
- A validation of our offline implementation of this generalized model for anisotropic materials against a ground truth (Sections 4 and 5).

2. The Isotropic Layered BSDF Model

In this section, we summarize the work of Belcour [2018] (see Figure 2). We refer readers to the original article for more details.

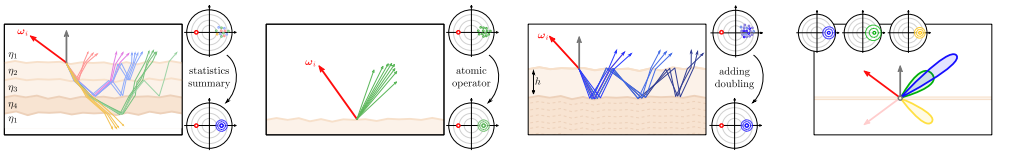
Mixture Model. In this model, light transport in the layered structure is approximated as a BSDF, ρ, that is a mixture of microfacet models with GGX normal distribution (Figure 2 (d)):

$$\rho(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \sum_k \rho_k \left(\boldsymbol{\omega}_k, \boldsymbol{\omega}_o; R_k, \alpha_k\right),$$

where ω_i and ω_o are, respectively, the incoming and outgoing directions, ρ_k is a GGX microfacet model with roughness α_k, incident direction $\boldsymbol{\omega}_k$, and directional albedo R_k that we will call a *BRDF lobe*:

$$\rho_k \left(\boldsymbol{\omega}_k, \boldsymbol{\omega}_o; R_k, \alpha_k\right) = \frac{R_k\, G(\boldsymbol{\omega}_k, \boldsymbol{\omega}_o)\, D(\boldsymbol{\omega}_h, \alpha_k)}{4\langle \boldsymbol{\omega}_k, \boldsymbol{n}\rangle \langle \boldsymbol{\omega}_o, \boldsymbol{n}\rangle},$$

where $\boldsymbol{\omega}_h = \frac{\boldsymbol{\omega}_k + \boldsymbol{\omega}_o}{||\boldsymbol{\omega}_k + \boldsymbol{\omega}_o||}$ is the half-vector direction, $D(\boldsymbol{\omega}_h)$ is the normal distribution function, and $G(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ is the shadowing/masking term.



(a) Global statistics (b) Atomic operator on statistics (c) Combining layers' statistics (d) Resulting mixture of BSDFs

Figure 2. (a) Belcour's model expresses the directional statistics (energy, mean, and variance) of a layered BSDF in the projected plane; (b) Instead of computing the complete transport, it tracks a statistical summary at each step. To do so, it decomposes light transport in different atomic operations (reflection, refraction, etc.); (c) Atomic operations are combined to evaluate multiple scattering between layers with a new *adding-doubling* algorithm working on statistics; (d) Finally, the model instantiates multiple BRDF lobes from those statistics to approximate the entire layered BSDF. (*Figure reproduced with permission.*)

The number of BRDF lobes is defined by the number of interfaces. The parameters of the kth BRDF lobe are derived from the statistics of the outgoing radiance of paths interacting up to the kth interface before exiting the surface (Figure 2 (c)). In this work, we concentrate on surface scattering only. However, our method is still compatible with forward scattering and absorption by participating media.

Statistical Analysis. The statistics studied by Belcour [2018] are the first three moments (i.e., energy, mean, and variance)[1] of directional radiance projected orthogonally on the unit disc. In this projected space, the BRDF lobe of an isotropic microfacet model such as GGX is close to radially symmetric. Thus, only the 1D radial variance of the lobe was studied. Mappings from moments (energy, mean, and variance) to the inputs of a BRDF lobe (directional albedo, incident direction, and roughness) are numerically computed for a microfacet BRDF model with GGX normal distribution [Walter et al. 2007]. In this setup the conversion from roughness to variance $\sigma = f(\alpha)$ and its inverse $\alpha = f^{-1}(\sigma)$ provides a way to retrieve the *equivalent roughness* of a GGX lobe for a given variance; the conversion from incident direction $\boldsymbol{\omega}_i$ to the first moment μ_i uses the orthographic projection $\mu_i = [\boldsymbol{\omega}_i.x, \boldsymbol{\omega}_i.y]$

The moments of all paths interacting up to the kth interface are computed using an *adding-doubling* algorithm [van de Hulst 1980] that takes as input and outputs zeroth-, first- and second-order moments (instead of radiance or irradiance for the original algorithm). This algorithm is made possible by the decomposition of light transport into atomic operators acting on the statistics of radiance. The different operators for surface interaction are summarized in Table 1. They require the knowledge of the roughness to variance mapping $\sigma = f(\alpha)$ and of the average reflected

	Rough Reflection	**Rough Refraction**
energy	$e_R = e_i \times r_{12}$	$e_T = e_i \times t_{12}$
mean	$\mu_R = -\mu_i$	$\mu_T = -\eta_{12}\,\mu_i$
variance	$\sigma_R = \sigma_i + f(\alpha_{12})$	$\sigma_T = \frac{\sigma_i}{\eta_{12}} + f(s \times \alpha_{12})$

Table 1. The atomic operators approximate the outgoing radiance's energy e, mean μ, and variance σ given an incident radiance's energy e_i, mean μ_i, and variance σ_i. For each statistic, the exponent indicates whether it has to be used for a reflected or transmitted lobe (for example σ^R for the variance of a reflected lobe). Those operators require the knowledge of the average reflected/transmitted energy, r_{12} and t_{12}, the index of refraction between the two medium η_{12}, the interface's roughness α_{12}, the fake refraction roughness scaling s, and the roughness to variance mapping, $\sigma = f(\alpha)$.

[1] We also use an abuse of notation and write σ to denote the variance.

(and transmitted) radiance at an interface:

$$r_{12} = \int_\Omega F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \frac{D_{\boldsymbol{\omega}_i}(\boldsymbol{\omega}_h) G_2(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)}{G_1(\boldsymbol{\omega}_i)} \mathrm{d}\boldsymbol{\omega}_h,$$

where $D_{\boldsymbol{\omega}_i}(\boldsymbol{\omega}_h)$ is the distribution of visible normals, G_2 and G_1 are the shadowing-masking functions [Heitz 2018], and $F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ is the Fresnel reflectance. For dielectric interfaces, the average transmitted energy is $t_{12} = 1 - r_{12}$.

Analytical Multiple Scattering. Since all operators are affine transformations of the statistics, multiple scattering between two interfaces has an analytical form. For example, the average energy of light reflected between two interfaces and finally transmitted through the top one (denoted e_R^∞) is approximated by

$$\begin{aligned} e_R^\infty &\simeq \sum_{i=0}^{\infty} t_{12}\, r_{23}\, \left[r_{21}\, r_{23}\right]^i\, t_{12}\, e_i, \\ &= \sum_{k=0}^{\infty} e_R^i, \end{aligned}$$

where e_R^i is the incident radiance (usually set to one). This approximation of e_R^∞ has an analytical form:

$$e_R^\infty \simeq \frac{r_{23}\, t_{12}^2}{\left[1 - r_{12}\, r_{23}\right]} e_i.$$

Similarly, the average transmitted radiance also has a closed form (see Listing 3 in Appendix A, lines 54–48). Note that those closed forms assume that light paths are not more angularly spread after two bounces on the same interface than after one bounce.

Under the same hypothesis, the average variance of light paths interacting with two interfaces before being transmitted through the top one is approximated using a weighted sum. Since the variance of a weighted sum of distribution is the weighted sum of the the distributions' variances, the average variance has a closed-form solution (see Listing 3, lines 69–71, and the original paper's Equation (38)).

Adding-Doubling. The *adding-doubling* algorithm leverages the fact that the average distribution of light interacting with two interfaces is equivalent to a reflection operator (the same argument holds for refraction). To compute the statistics of radiance reflected and refracted by a stack of layers, the adding-doubling algorithm iteratively stacks interfaces (starting from the top to the bottom in our case) and tracks the reflected and transmitted statistics for light coming from above the surface and light coming from below using the previously mentioned closed form (see Listing 3 in Appendix A). With these building blocks, we can build any layered material configuration.

3. The Anisotropic Layered BSDF Model

We extend the statistical model described in the previous section to support anisotropic interfaces. Our layered anisotropic BSDF model is expressed as a weighted sum of anisotropic GGX BRDF lobes:

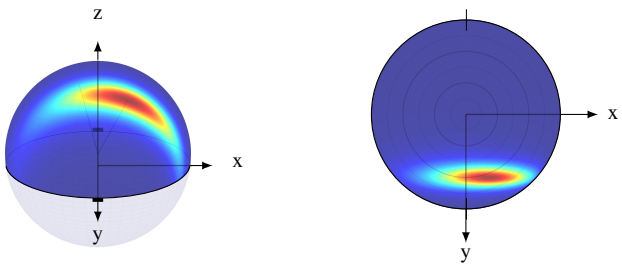
$$\rho(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \sum_k \rho_k \left(\boldsymbol{\omega}_k, \boldsymbol{\omega}_o; R_k, \boldsymbol{\alpha_k}\right),$$

where $\boldsymbol{\alpha}_k = (\alpha_{k,x}, \alpha_{k,y})$ is the anisotropic roughness for the GGX microfacet model ρ_k. We use the adding-doubling strategy on the statistics of the incident and outgoing radiance to estimate those parameters. However, the adding-doubling algorithm is no longer performed on a scalar value, but on a vector of two components. This is possible since the BRDF lobes are axis-aligned (see Section 3.1)
In the following sections, we explain:

- Our analysis of BRDF lobes in the tangent plane (Section 3.1);
- How we estimate the outgoing average energy from an anisotropic BSDF lobe to estimate the outgoing energy and mix the variances (Section 3.2);
- How to convert an anisotropic roughness to a couple of *linear variance* using our new mapping $(\sigma_x, \sigma_y) = f(\alpha_x, \alpha_y)$ (Section 3.3).

3.1. Analysis of Anisotropic BRDFs

We studied the shape of anisotropic BRDF lobes when projected in the unit disk (see Figure 3). For the GGX microfacet model, we found that the projected distribution was aligned with the anisotropy axis (see Figure 4 and our supplemental material). This insight enables us to treat the variance along each axis independently. Instead of performing the adding-doubling algorithm on a scalar variance, we perform it on a two-component variance vector (one component per anisotropy axis). See Appendix A for a detailed implementation of the algorithm.



(a) Spherical space (b) Projective space

Figure 3. To study the statistics of the BRDF lobe resulting from a GGX distribution (a) in the spherical domain, we use the space of directions projected onto the tangent plane (b).

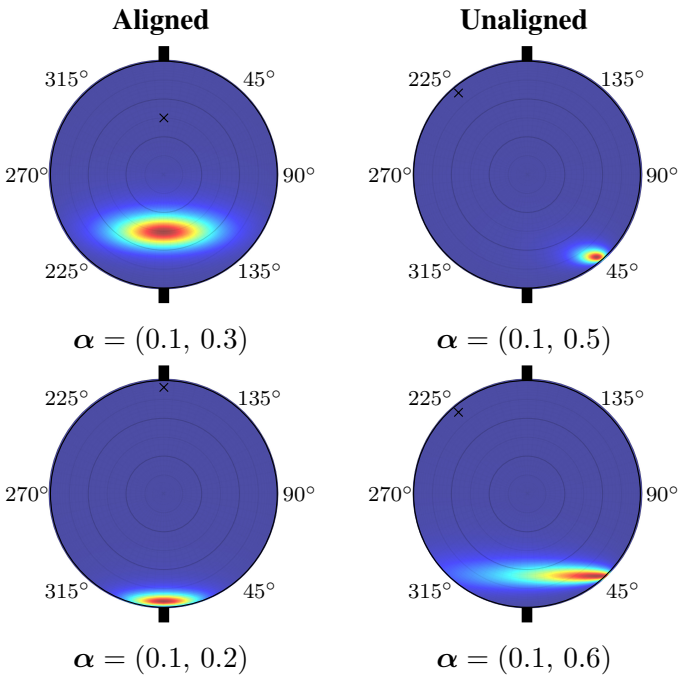


Figure 4. The shape of the projected BRDF lobe with a GGX distribution remains axis-aligned whatever the incident direction $\boldsymbol{\omega}_i = (\theta,\phi)$ (indicated by the cross) or roughness $\boldsymbol{\alpha} = (\alpha_x, \alpha_y)$. As shown, both when the incident direction is aligned with the tangent frame (left) or unaligned (right), the outgoing lobe is still axis-aligned with the tangent frame.

A dependence between the anisotropy axes is still maintained by the estimation of the average reflected/transmitted energy in an anisotropic BRDF lobe (see `Evaluate_SchlickFresnelAniso` function call, line 42 of Listing 3), and we detail it in Section 3.2. The last element not described is the conversion from roughness to variance (see `roughnessToVariance` and `varianceToRoughness`) function calls in Listing 3), and we describe it in Section 3.3).

3.2. Average Energy for One Interaction

In the isotropic case, the average reflectance, r_{12}, is tabulated on a 4D regular grid. However, Bati et al. [2019] found that for offline rendering, this pre-integration provided little benefit. We followed their finding and removed both the average reflectance and correction term for total internal reflection in our offline implementation. However, in our real-time implementation, we found that the pre-integrated FGD term has a non-negligible visual impact for the first lobe. In game engines, the Fresnel reflectance is approximated using Schlick's Fresnel [1994]. This approximation separates the Fresnel in two terms, which simplifies its pre-integration with a split-sum [Karis 2013]. Including the effect of anisotropy in such a model would

require moving the 2D split-sum texture to a 4D one, accounting for the view vector's azimuth and the second roughness. We introduce a method that does not require additional storage. We start from Schlick's Fresnel formulation,

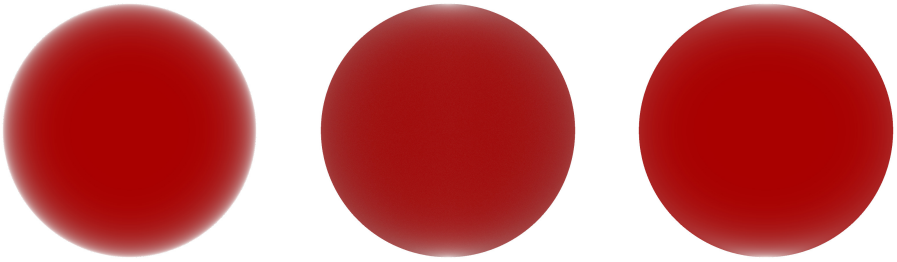
$$F\left(\cos(\theta_D)\right) = R_0 + (1 - R_0)\left(1 - \cos(\theta_D)^5\right),$$

where θ_D is the difference vector in the half-angle parametrization and R_0 is the reflectance at normal incidence defined by the refractive index. The directional albedo is thus

$$r_{12}(\boldsymbol{\omega}_i) = \int_\Omega F\left(\langle \boldsymbol{\omega}_i, \boldsymbol{\omega}_h \rangle\right) \frac{D_{\boldsymbol{\omega}_i}(\boldsymbol{\omega}_h) G_2(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)}{G_1(\boldsymbol{\omega}_i)} \mathrm{d}\boldsymbol{\omega}_h,$$

where $G(.,.)$ is the shadowing/masking function and $D(.)$ is the normal distribution function. Instead of searching for a separable form for $r_{12}(\boldsymbol{\omega}_i)$, which would be difficult due to the integral over solid angles (remember that reflected radiance is separable in the projected disc space), we found a solution that only required scaling the input $\boldsymbol{\omega}_i$ of the isotropic $r_{12}(\boldsymbol{\omega}_i)$.

Anisotropic Fetch We found that the shape of the average reflectance of Schlick's Fresnel with an anisotropic BSDF lobe was close to a stretching of the average reflectance of an isotropic BSDF lobe (see Figures 5 (a) and (b)). Hence, to evaluate the anisotropic average reflectance, we stretch the incident direction by the difference in roughness: $\boldsymbol{\omega}'_x = \frac{1}{1+\Delta\alpha_x^2}\boldsymbol{\omega}_x$, and $\boldsymbol{\omega}'_y = \frac{1}{1+\Delta\alpha_y^2}\boldsymbol{\omega}_y$; and obtain ω_z from $\boldsymbol{\omega}'_x$ and $\boldsymbol{\omega}'_y$). Listing 1 shows our implementation. With this method, we only need to store the split-sum texture of Karis and modify the fetch function to handle the anisotropy. Figure 5 shows how our method better reproduces the anisotropic effect of average Fresnel in a white-furnace environment. Our method extends to the general form of the pre-integrated Fresnel reflectance. Please refer to our supplemental material for an interactive validation.



(a) Karis [2013] (b) Reference (c) Ours

Figure 5. Preintegrating the Fresnel term using the split-sum approximation (a) fails to account for the anisotropy of the lobe (b). In this white-furnace setting with $\boldsymbol{\alpha} = (0, 0.5)$, our solution (c) correctly captures the anisotropic shape of the Fresnel term.

```
void Evaluate_SchlickFresnelAniso(vec3 wi, vec3 F0, vec2 alpha,
                                  out vec3 R, out vec3 T) {

    // Scale the incident direction
    float min_axy = min(alpha.x, alpha.a.y);
    float delta_x = alpha.x - min_axy;
    float delta_y = alpha.y - min_axy;
    wi.xy = vec2(wi.x/(1.0+delta_x*delta_x), wi.y/(1.0+delta_y*delta_y));
    wi.z  = sqrt(1.0 - dot(wi.xy, wi.xy));

    // Eval the part that is used for the split integral
    vec2 splitSum = textureLod(t_splitSum, vec2(wi.z, min_axy), 0.0).xy;

    // Return the result
    // If the interface is not transmissive, do not compute T
    R = vec3(1)*splitSum.x + F0*splitSum.y;
    T = 1.0 - R;
}
```

Listing 1. We evaluate the anisotropic r_{12} term from the classical split-sum element by scaling the incident direction.

Note that this method is not restricted to the layered BSDF model and could be applied to surface shading with anisotropic microfacet models. In our implementation, we did not look at the problem of multiple scattering in the micro-surface. However, the method of Turquin [2019] could be used to rescale the missing energy.

3.3. A New Linear Mapping

Due to the shape of the GGX microfacet distribution (notably its tail), the reflection operator on variance is inaccurate. More precisely, it tends to over-estimate the variance and leads to over-blurring. To compensate, Belcour defined a corrected variance space, the *linear variance* space. The idea is to modify the conversion from roughness to variance to correct for this over-estimation.

Say we have an invertible function f that converts an input roughness into variance:

$$\boldsymbol{\sigma} = f(\boldsymbol{\alpha}) \text{, and } \boldsymbol{\alpha} = f^{-1}(\boldsymbol{\sigma}).$$

Ideally, if the incident radiance were distributed as a Gaussian and the reflection operator were a convolution by a Gaussian kernel, the variance of the outgoing reflected radiance would be

$$\boldsymbol{\sigma}_o = f(\boldsymbol{\alpha}) + \boldsymbol{\sigma}_i,$$

where α is the roughness of the surface and $\boldsymbol{\sigma}_i$ (respectively, $\boldsymbol{\sigma}_o$) is the variance of the incident (respetively, outgoing) radiance. Thus, the variance after two bounces of a Dirac incident lighting would be

$$\boldsymbol{\sigma}_{12} = f(\boldsymbol{\alpha}_1) + f(\boldsymbol{\alpha}_2). \tag{1}$$

In this work, we generate this linear variance space using non-linear fitting of parametric curves f and f^{-1}. We do so by exploiting Equation (1) and computing numerically the variance of a single reflection for a varying anisotropic roughness $\boldsymbol{\sigma} = f_d(\boldsymbol{\alpha})$ (see Figure 6 (a)). By numerically finding the inverse of this function, we obtain $\boldsymbol{\alpha} = f_d^{-1}(\boldsymbol{\sigma})$, and we can use this mapping to extract the single scattering roughness that correspond to a given variance (such variance could come from light scattered multiple times). We also compute the variance of outgoing radiance after two bounces on plane parallel surfaces $\boldsymbol{\sigma}_{12} = g_d(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2)$. Finally, we optimize our parametric model to fit the following equality:

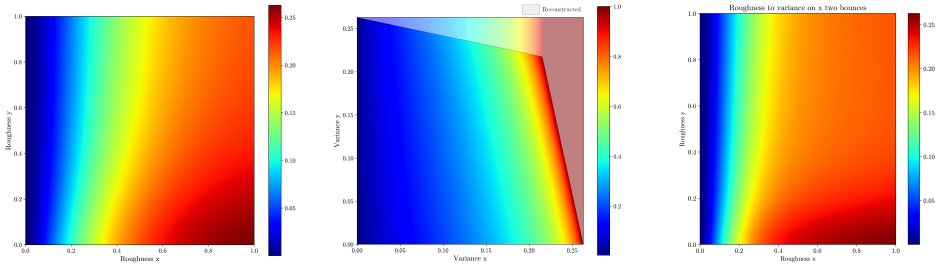
$$f^{-1}\left(g_d\left(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2\right)\right) = f^{-1}\left(f\left(\boldsymbol{\alpha}_1\right) + f\left(\boldsymbol{\alpha}_2\right)\right).$$

We found out that the roughness α_y has little influence on the variance σ_x for low values of α_x. It only affected σ_x for high roughness (see Figure 6 (a)). By symmetry, the same can be observed for σ_y with low values of α_y. This indicates that the mapping can be approximated in a simpler form with 1D functions per anisotropy axis that decorrelate the dimensions:

$$\begin{aligned} f(\boldsymbol{\alpha}) &= (f_1(\alpha_x), f_1(\alpha_y)) \\ f^{-1}(\boldsymbol{\sigma}) &= \left(f_1^{-1}(\sigma_x), f_1^{-1}(\sigma_y)\right). \end{aligned}$$

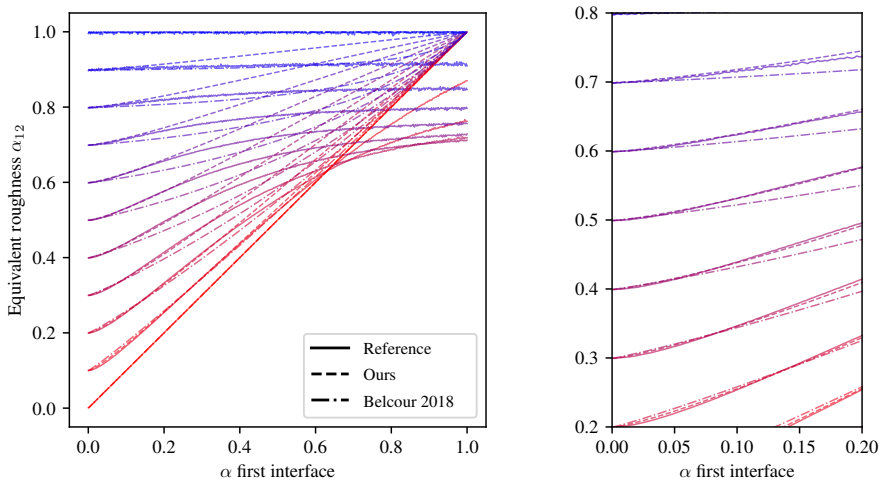
We used a log sigmoid and let the optimizer choose its parameters:

$$\begin{aligned} f_1(x) &= \log\left(1 + \frac{b\, x^a}{1 - x^a}\right) \\ f_1^{-1}(y) &= \left(\frac{e^y - 1}{e^y - 1 + b}\right)^{\frac{1}{a}}. \end{aligned}$$



(a) $\sigma_x = f_d(\alpha_x, \alpha_y)$ (b) $\alpha_x = f_d^{-1}(\sigma_x, \sigma_y)$ (c) $\boldsymbol{\alpha}_{12} = f_d^{-1}\left(g_d\left(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2\right)\right)$

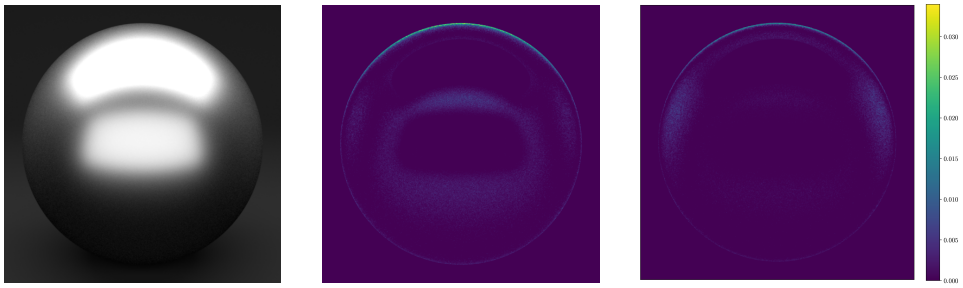
Figure 6. Our linear mapping f converts an input anisotropic roughness $\boldsymbol{\alpha}$ to a pair of variances $\boldsymbol{\sigma} = (\sigma_x, \sigma_y)$. To build f, we numerically computed f_d (a) and its inverse mapping f_d^{-1} that converts a variance to its associated roughness (b). For this latter function, we had to extrapolate for the (σ_x, σ_y) values that were not covered by f_d (highlighted in gray). With that function, we can display the *equivalent roughness* of radiance after two bounces (c). We optimized both function f and f^{-1} with respect to the data for the two-bounces case g_d (c).



(a) Equivalent roughness grid (b) Zoom

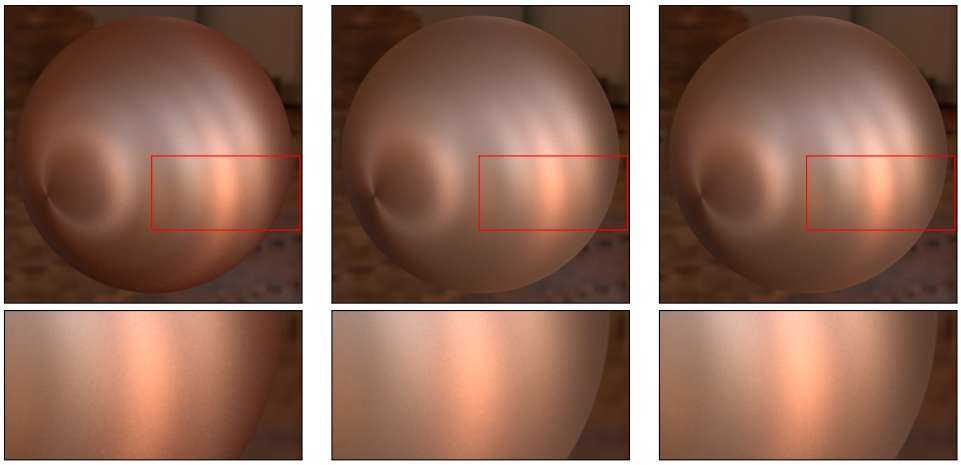
Figure 7. In this plot, we show the equivalent roughness after two bounces on isotropic surfaces at normal incidence. The abscissa denote the first roughness of the first bounce and the color of the curve (red for zero and blue for one) the roughness of the second bounce. Our new linear mapping function better fits the ground truth data for low roughnesses compared to the one of Belcour (see zoom-in one the right). However, both methods fail to correctly approximate high roughness (see (a) on the left) as they converge to one for any second interface roughness.

We found that using $a \simeq 1.29$ and $b \simeq 1.31$ gives a closer match to the reference curve of equivalent roughness after two bounces than Belcour's fit for the isotropic case (see Figure 7) and provides a lower error compared to the ground truth when used in the adding-doubling scheme (see Figure 8). Thus, we recommend using our



(a) Reference (b) [Belcour 2018] (c) Ours

Figure 8. In this scene, our fit of the linear mapping (c) reduces the error to the ground truth compared to Belcour's fit (b). We used $\boldsymbol{\alpha}_1 = (0.1, 0.1), \eta_1 = 1.5$ for the first layer and $\boldsymbol{\alpha}_2 = (0.1, 0.1), \eta_2 = 0.0, \kappa_2 = 1.0$ for the second layer.



(a) Reference (b) 2D fit (c) Decorrelated fit

Figure 9. Our decorrelated fit (c) produces perceptually close rendering compared to a reference (a). Using a more complex fit (b) improves accuracy but the gain can be hard to notice.

fit instead, even for the isotropic case. Note that we computed f_d and g_d using a normal incident direction with unit energy.

We tested the alternative of fitting a 2D function and its inverse (see Appendix B for details). We found that the decorrelated fit was perceptually as close to the reference as the 2D fit and that the increased complexity and evaluation cost of this latter method was hard to justify (see Figure 9). Our implementation of this mapping is provided in Listing 2.

```
#define CONST_A 1.28809776
#define CONST_B 1.31699416

vec2 roughnessToVariance(vec2 a)
{
    vec2 aPow = clamp(pow(a, vec2(CONST_A)), 0.0, 0.99999);
    return log(vec2(1.0) + (vec2(CONST_B) * aPow) / (vec2(1.0) - aPow));
}

vec2 varianceToRoughness(vec2 v)
{
    vec2 c = exp(v) - vec2(1.0);
    return pow(c / (c + vec2(CONST_B)), vec2(1.0 / CONST_A));
}
```

Listing 2. We computed a new conversion function from roughness to variance and back.

4. Results

In this section, we present the results of and validation for using our layered BSDF model. First, we implemented it as a Mitsuba [Jakob 2010] plugin to validate it against a ground truth (Section 4.1). Second, we implemented it in the Unity engine and

present our real-time results (Section 4.2). All our computation and timings were run on an Intel i7-6900K + Nvidia Geforce GTX 1060 GPU.

4.1. Offline Results

We extensively compared our approximative model against a ground truth. Our ground truth is computed as a stochastic evaluation of light transport in the layered structure. Please refer to our interactive supplemental material for more details and results. Our model is visually close to the reference for low to moderate ($\alpha \simeq 0.3$) roughnesses.

Figures 10 and 11 display a subset of our validation material. In Figure 10, we fix an isotropic top layer of $\boldsymbol{\alpha} = (0.05, 0.05)$ and vary the bottom layer's roughness.

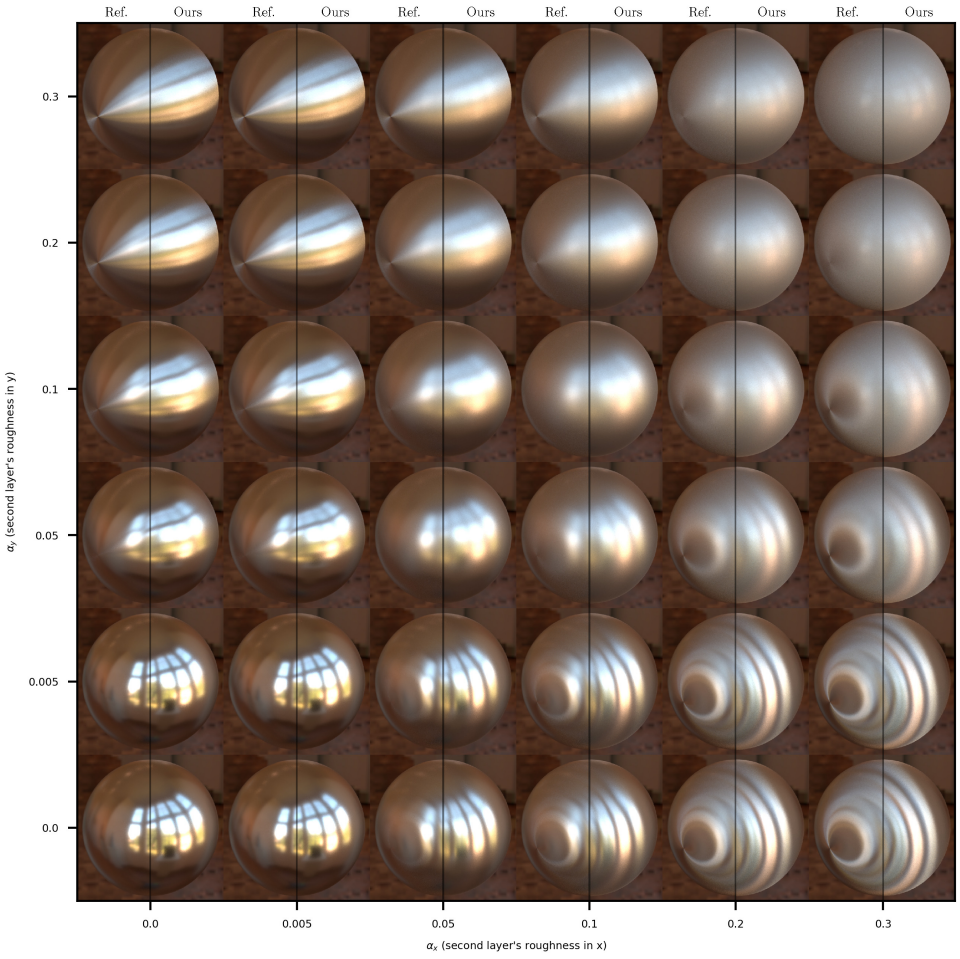


Figure 10. A comparison between the reference and our decorrelated model with a fixed roughness $\alpha_1 = (0.05, 0.05)$ for the dielectric layer and a second conductor layer with varying roughness

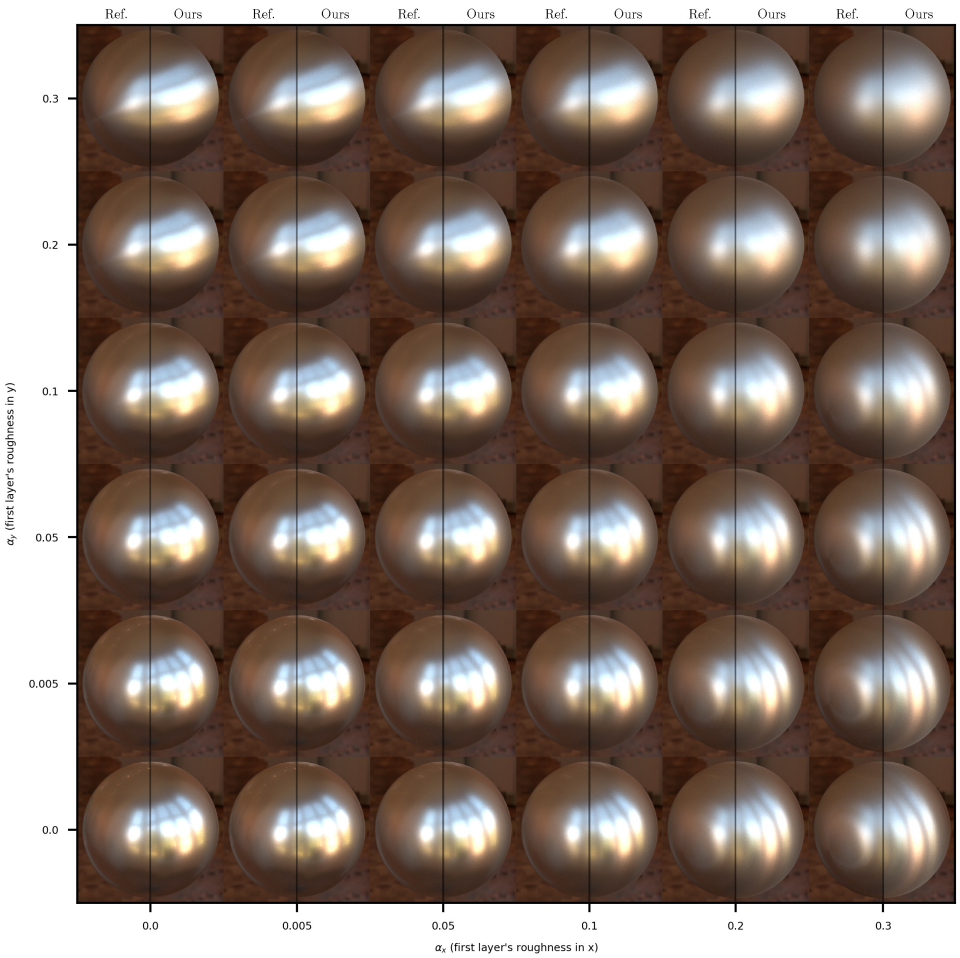


Figure 11. A comparison between the reference and our decorrelated model with a varying roughness for the first dielectric layer and a second conductor layer with fixed roughness $\alpha_1 = (0.05, 0.05)$

In Figure 11, we fix an isotropic bottom layer of $\boldsymbol{\alpha} = (0.05, 0.05)$ and vary the top layer's roughness. For both examples, the top layer IOR is $\eta_1 = 1.5$ and the bottom complex IOR is $\eta_2 = 0.0, \kappa_2 = 1.0$.

Timings. We compared the running time of our method with the one of Belcour. We list our timings in Table 2.

	Area Light (512spp)	Teaser (512spp)
[Belcour 2018]	12.1s	5.6m
Ours	15.2s	6.3m

Table 2. We measured the difference in rendering time between using the isotropic model of Belcour and our anisotropic extension. For the Area Light scene, we set the roughnesses to zero to measure the difference in the computation of the adding-doubling method.

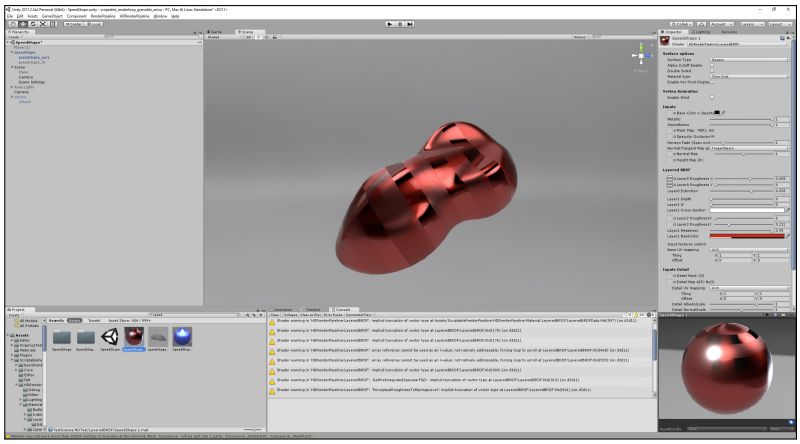


Figure 12. A simple textured asset rendered in Unity using our implementation.

4.2. Real-Time Results

We implemented our anisotropic layered material model in the Unity engine, on top of the high-definition rendering pipeline (HDRP) (see Figure 12). We made minimal changes to the implementation, but found out that the use of our model will be limited by how engines render anisotropic BRDF models with image based lighting. The HDRP uses the method of Revie [2011] to fake the preintegration of environment lighting with anisotropic BRDF lobes. However, with this method two lobes with the same mean direction but different roughness will not appear aligned. While this is not a big deal when one has to render one BRDF lobe, it impacts the realism of our model that consists of many BRDF lobes. To overcome this, we use the same technique as Yamaguchi et al. [2019] and use Monte-Carlo integration. This greatly diminishes the performance of the shading model and impedes its use in production: for 128 Monte-Carlo samples, our method takes between 10ms to 42ms to render a frame at 720p, depending on the coherence of the texture fetches.

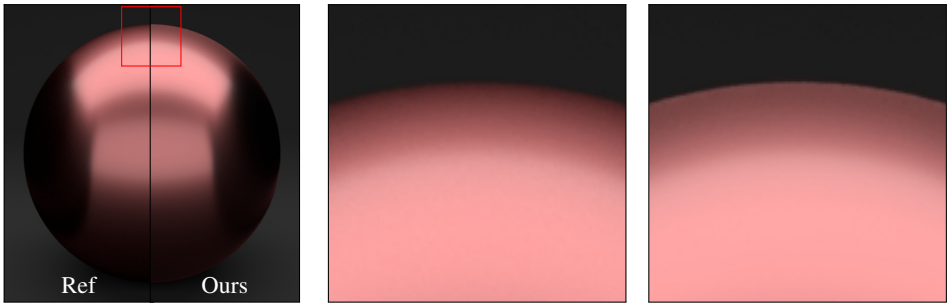
Anisotropic IBL Preintegration. There is no working solution so far to pre-filter cube-map or envmap textures with respect to anisotropic BRDF lobes, nor there is a working solution to shade area-light sources with anisotropic highlights. Thus, there is a reduced applicability of anisotropic layered materials for real-time rendering. An interesting research avenue lies in the ability to perform such evaluations.

5. Limitations

High Roughnesses. In Figure 10 we can observe the difference with a ground-truth; for very low roughness our model closely approximates the reference even at grazing angles. At higher roughness, our model slightly overestimates the roughness at graz-

ing angles. This is due to the fact that our Fresnel is only accurate at normal incidence, since our second-order statistic computation for the first and second bounce assumes a normal incidence. A more accurate model would introduce a correction factor that accounts for these changes at grazing angles, but the complexity of a brute-force calculation of all the parameters would result in a high-dimensional function not suited for a fast evaluation.

Grazing Angles. When considering a varying top layer as in Figure 11, we observe that this effect is even stronger since the conductor layer almost acts as a mirror and most energy is then scattered by the dielectric layer. A clear depiction of this phenomena can be seen in Figure 13. For more results and an interactive viewer for different scenes please refer to our supplemental material.

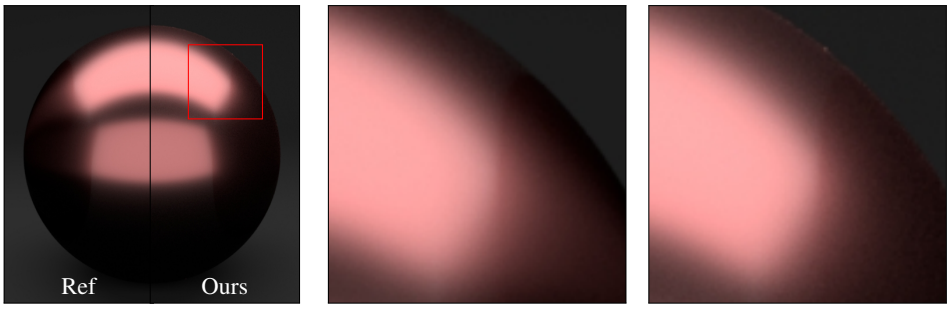


(a) Area Light (b) Crop - Reference (c) Crop - Ours

Figure 13. Our method differs from the reference at grazing angles. This is due to the correlation between total internal reflection (TIR) and our precomputed average reflectance for one bounce. Since we do not account for it, our model overblurs the reflectance at grazing angles. We used $\boldsymbol{\alpha_1} = (0.05, 0.05)$ for the top layer and $\boldsymbol{\alpha_2} = (0.1, 0.0)$ for the bottom layer.

Opposing Anisotropies. Another limitation of the statistical model is that it cannot represent layers with opposing anisotropies well. A material with two layers, each with strong anisotropy along a different axis (say $(1, 0)$ for the first layer and $(0, 1)$ for the second layer), will result in a diamond-shaped highlight that is not possible to represent with a GGX normal distribution. We showcase this effect in Figure 14.

Anisotropy Axis. In our work, we assumed that the different layers shared the same anisotropy axis. Working with different anisotropy axes would require tracking the full 2×2 covariance matrix in the adding-doubling process as Yamaguchi et al. [2019] does. Our method could, without restriction, also be extended to do so.



(a) Area Light (b) Crop - Reference (c) Crop - Ours

Figure 14. When the anisotropy axes are opposed between the two layers, a slightly stronger horizontal blur can be seen in the reference. In this case we have the following configuration: $\boldsymbol{\alpha_1} = (0.1, 0.0)$ and $\boldsymbol{\alpha_2} = (0.0, 0.05)$

6. Conclusion

We presented an extension of the layered BRDF model of Belcour [2018] to handle anisotropic surfaces. We built on the analysis that when projected on the tangent plane, BRDF lobes are closely approximated by ellipsoidal-shaped distributions aligned with the tangent frame axis. Based on that observation, we re-used the adding-doubling method on a vector of variances and treated each anisotropy axis independently. Doing so, we improved the conversion of roughness to variance to better fit the isotropic and anisotropic case. We improved the split-sum form of Karis [2013] to handle anisotropic BRDF lobes.

Acknowledgements

Thanks to Merlin Nimier-David for the interactive viewer framework used in our supplemental material.

A. Anisotropic Adding-Doubling Code

```
void ComputeBsdfLobes(in vec3 wi, in BsdfLayer[NUM_LAYERS] layers, out BsdfLobe[NUM_LAYERS] lobes)
{
    /* Init all layers to zero */
    for(int i=0; i<NUM_LAYERS; ++i) {
        lobes[i].wi = wi;
        lobes[i].R  = vec3(0);
        lobes[i].a  = vec2(0);
    }

    /* Define the different global terms for the adding-doubling */
    vec3 R0i = vec3(0.0), Ri0 = vec3(0.0), T0i = vec3(1.0), Ti0 = vec3(1.0);
    vec2 s_r0i = vec2(0.0), s_ri0 = vec2(0.0), s_t0i = vec2(0.0), s_ti0 = vec2(0.0);
    float j0i = 1.0, ji0 = 1.0;

    /* Iterate on all the layers and apply the adding-doubling equations */
    for(int i=0; i<NUM_LAYERS; ++i)
    {
        // Evaluate the IOR ratio
        float eta = abs(1.0/layers[i].n);

```

```
            /* Evaluate Snell law */
            vec3 wt = -refract(-wi, vec3(0,0,1), 1.0/layers[0].n);

            /* Evaluate the variance of the reflected lobe and transmitted lobe
             * for a dirac impulse. */
            vec2 s_r12 = roughnessToVariance(layers[i].a);
            vec2 s_r21 = s_r12;

            vec2 s_t12, s_t21; float j12=1.0, j21=1.0;
            if(wt.z >= 0.0) {
                const vec2 alpha_scale = layers[i].a * 0.5 * abs(wt.z * eta - wi.z);
                s_t12 = roughnessToVariance(alpha_scale / (wt.z * eta));
                s_t21 = roughnessToVariance(alpha_scale / (wi.z / eta));

                // Scale due to the interface
                j12 = (wt.z / wi.z) * eta;
                j21 = 1.0 / j12;
            }

            /* Evaluate the reflection coefficent R12 and the transmission coefficient T12 */
            vec3 R12, R21, T12, T21;
            Evaluate_SchlickFresnelAniso(wi, layers[i], R12, T12);
            R21 = R12; T21 = T12;


            /* Multiple scattering forms */
            const vec3  denom = (vec3(1.0) - Ri0 * R12);
            const float d_avg = average(denom);
            const vec3  m_R0i = (T0i * R12 * Ti0) / denom;
            const vec3  m_Ri0 = (T21 * Ri0 * T12) / denom;
            const vec3  m_Rr  = (Ri0 * R12) / denom;

            // Evaluate the adding operator on the energy
            const vec3 e_R0i = R0i + m_R0i;
            const vec3 e_T0i = (T0i * T12) / denom;
            const vec3 e_Ri0 = R21 + m_Ri0;
            const vec3 e_Ti0 = (T21 * Ti0) / denom;

            // Scalar forms for the spectral quantities
            const float r0i   = average( R0i );
            const float e_r0i = average( e_R0i );
            const float e_ri0 = average( e_Ri0 );
            const float m_r0i = average( m_R0i );
            const float m_ri0 = average( m_Ri0 );
            const float m_rr  = average( m_Rr );
            const float r21   = average( R21 );

            /* Evaluate the adding operator on the normalized variance */
            vec2 _s_r0i = (r0i * s_r0i + m_r0i * (s_ti0 + j0i * (s_t0i + s_r12 + m_rr * (s_r12 + s_ri0)
    )));
            vec2 _s_t0i = j12 * s_t0i + s_t12 + j12 * (s_r12 + s_ri0) * m_rr;
            vec2 _s_ri0 = (r21 * s_r21 + m_ri0 * (s_t12 + j12 * (s_t21 + s_ri0 + m_rr * (s_r12 + s_ri0)
    )));
            vec2 _s_ti0 = ji0 * s_t21 + s_ti0 + ji0 * (s_r12 + s_ri0) * m_rr;
            _s_r0i = (e_r0i > 0.0) ? _s_r0i / e_r0i : vec2(0.0);
            _s_ri0 = (e_ri0 > 0.0) ? _s_ri0 / e_ri0 : vec2(0.0);

            /* Store the mean reflectance and roughness */
            lobes[i].R = m_R0i;
            lobes[i].a = varianceToRoughness( s_ti0 + j0i * (s_t0i + s_r12 + m_rr * (s_r12 + s_ri0)) );

            /* Update energy */
            R0i = e_R0i;
            T0i = e_T0i;
            Ri0 = e_Ri0;
            Ti0 = e_Ti0;

            /* Update mean */
            wi = wt;

            /* Update variance */
            s_r0i = _s_r0i;
            s_t0i = _s_t0i;
            s_ri0 = _s_ri0;
            s_ti0 = _s_ti0;

            /* Update jacobian */
```

```
            j0i *= j12;
            ji0 *= j21;

            /* Early exit in case the layer does not transmit light */
            if(layers[i].n < 0) return;
        }
    }

```

Listing 3. The adding-doubling algorithm takes as input a set of layers and a view-vector configuration and outputs a set of BRDF lobes that approximate the correct light transport.

B. Linear Mapping: A 2D Fit

In this section, we present the form used for our 2D fit (see Figure 9). Due to symmetry of the inputs, we choose to use the same 2D function for both dimensions by swapping the inputs:

$$f(\boldsymbol{\alpha}) = \left[f_x(\alpha_x, \alpha_y), f_x(\alpha_y, \alpha_x)\right],$$
$$f^{-1}(\boldsymbol{\alpha}) = \left[f_x^{-1}(\alpha_x, \alpha_y), f_x^{-1}(\alpha_y, \alpha_x)\right].$$

In practice, we optimized

$$f_x(\alpha_x, \alpha_y) = \alpha_x \log\left(1 + \frac{b\alpha_x^a}{1-\alpha_x^a}\right) \exp\left(c\left(\alpha_x - \alpha_y\right)^2 + d\left(\alpha_x - \alpha_y\right)\right),$$

with $a = 0.0134517769$, $b = 0.0123629536$, $c = 0.493769777$, and $d = 0.225934414$ for the evaluation of roughness to variance. And

$$f_x^{-1}(\sigma_x, \sigma_y) = \left[\frac{1}{1 + \frac{b}{e^{\sigma_x}-1}}\right]^{1/a} \left[\frac{1}{1 + \frac{c}{e^{\sigma_y}-1}}\right]^{1/d} \exp\left(e\left[\sigma_x - \sigma_y\right]\right),$$

with $a = 1.38609881$, $b = 0.609910674$, $c = 100.608653$, $d = 64.1513931$, and $e = 10^{-05}$ for the evaluation of variance to roughness.

Index of Supplemental Materials

We provide the following supplemental materials:

- Mitsuba plugins containing both our layered material and our reference code can be found at `jcgt.org/published/0009/02/03/mitsuba_supplemental.zip`. We also provide the Area Light scene.
- An HTML webpage with interactive results can be found at `jcgt.org/published/0009/02/03/html_supplemental.zip`. . This archive is 4Gb and contains a couple of HTML pages with results. To use this page, we recommend launchng a simple HTTP server at the root of the archive (see README for more details).

References

BATI, M., PACANOWSKI, R., AND BARLA, P. 2019. Comparative study of layered material models. In *EGSR's Workshop on Material Appearance Modeling*. Eurographics Associ-

ation, Aire-la-Ville, Switzerland. URL: https://hal.archives-ouvertes.fr/hal-02184562/document. 43

BELCOUR, L. 2018. Efficient rendering of layered materials using an atomic decomposition with statistical operators. *ACM Trans. Graph. (2018)*. URL: https://dl.acm.org/doi/10.1145/3197517.3201289. 37, 38, 39, 40, 47, 50, 53

BURLEY, B. 2012. Physically-based shading at Disney. In *ACM SIGGRAPH - Practical Physically Based Shading in Film and Game Production Course*, vol. 2012. ACM, New York, NY, USA, 1–7. URL: https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf. 38

GUO, Y., HASAN, M., AND ZHAO, S. 2018. Position-free monte carlo simulation for arbitrary layered bsdfs. *ACM Trans. Graph. 37*, 279:1–279:14. URL: https://doi.org/10.1145/3272127.3275053. 38

HEITZ, E. 2018. Sampling the ggx distribution of visible normals. *Journal of Computer Graphics Techniques (JCGT) 7*, 4 (November), 1–13. URL: http://jcgt.org/published/0007/04/01/. 41

JAKOB, W., D'EON, E., AND MARSCHNER, S. 2014. A comprehensive framework for rendering layered materials. *ACM Transactions on Graphics 33*, 114. URL: https://doi.org/10.1145/2601097.2601139. 38

JAKOB, W., 2010. Mitsuba renderer. URL: http://www.mitsuba-renderer.org. 48

KARIS, B. 2013. Real shading in unreal engine 4. In *Physically Based Shading in Theory and Practice - SIGGRAPH Courses*. ACM, New York, NY, USA. URL: https://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf. 43, 44, 53

REVIE, D. 2011. Implementing fur using deferred shading. In *GPU Pro 2*, W. Engel, Ed. A K Peters, Ltd., Natick, MA, USA, ch. 2, 57. URL: https://dl.acm.org/doi/book/10.5555/1996203. 51

SCHLICK, C. 1994. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum 13*, 3, 233–246. URL: https://doi.org/10.1111/1467-8659.1330233. 43

TURQUIN, E., 2019. Practical multiple scattering compensation for microfacet models. URL: https://blog.selfshadow.com/publications/turquin/ms_comp_final.pdf. 45

VAN DE HULST, H. C. 1980. *Multiple Light Scattering*, vol. 1. Academic Press, New York. 40

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Eurographics Symposium on Rendering*. Eurographics Association, Aire-la-Ville, Switzerland. URL: https://dl.acm.org/doi/10.5555/2383847.2383874. 40

YAMAGUCHI, T., YATAGAWA, T., TOKUYOSHI, Y., AND MORISHIMA, S. 2019. Real-time Rendering of Layered Materials with Anisotorpic Normal Distributions. In *SIGGRAPH Asia 2019, Technical Brief*. ACM, New York, NY, USA. URL: https://dl.acm.org/doi/10.1145/3355088.3365165. 38, 51, 52

ZELTNER, T., AND JAKOB, W. 2018. The layer laboratory: A calculus for additive and subtractive composition of anisotropic surface reflectance. *Transactions on Graphics (Proceedings of SIGGRAPH) 37*, 4 (July), 74:1–74:14. URL: https://dl.acm.org/doi/10.1145/3197517.3201321. 38

Author Contact Information

Philippe Weier
École Polytechnique Fédérale de Lausanne
philippe.weier@epfl.ch
https://weiphil.github.io/portfolio

Laurent Belcour
Unity Technologies
laurent@unity3d.com
http://belcour.github.io/blog

Weier and Belcour, Rendering Layered Materials with Anisotropic Interfaces, *Journal of Computer Graphics Techniques (JCGT)*, vol. 9, no. 2, 37–57, 2020
http://jcgt.org/published/0009/02/03/

Received:	2019-11-05		
Recommended:	2020-03-10	Corresponding Editor:	Naty Hoffman
Published:	2020-06-20	Editor-in-Chief:	Marc Olano