Bringing Linearly Transformed Cosines to Anisotropic GGX

Aakash KT
CVIT, KCIS, IIIT-Hyderabad

Eric Heitz
Unity Technologies

Jonathan Dupuy
Unity Technologies

P. J. Narayanan
CVIT, KCIS, IIIT-Hyderabad

Figure 1: Real-time area lighting with Linearly Transformed Cosines (LTCs) in a commercial game engine. ***Our LTC approximation for anisotropic materials takes 0.7 ms at 1080p resolution on an NVIDIA GeForce RTX 2080 GPU.***

ABSTRACT

Linearly Transformed Cosines (LTCs) are a family of distributions that are used for real-time area-light shading thanks to their analytic integration properties. Modern game engines use an LTC approximation of the ubiquitous GGX model, but currently this approximation only exists for isotropic GGX and thus anisotropic GGX is not supported. While the higher dimensionality presents a challenge in itself, we show that several additional problems arise when fitting, post-processing, storing, and interpolating LTCs in the anisotropic case. Each of these operations must be done carefully to avoid rendering artifacts. We find robust solutions for each operation by introducing and exploiting invariance properties of LTCs. As a result, we obtain a small 8^4 look-up table that provides a plausible and artifact-free LTC approximation to anisotropic GGX and brings it to real-time area-light shading.

ACM Reference Format:
Aakash KT, Eric Heitz, Jonathan Dupuy, and P. J. Narayanan. 2022. Bringing Linearly Transformed Cosines to Anisotropic GGX. In *Proceedings of SIGGRAPH I3D (I3D'22).* ACM, New York, NY, USA, 9 pages.

Acknowledgements. Thanks to Thomas Deliot for making a prototype in the Unity engine and to Stephen Hill for his feedback. Aakash KT was funded by the "Kohli Fellowhip" of KCIS.

1 INTRODUCTION

Today's physically based shading models are largely based on the GGX *Bidirectional Reflectance Distribution Function* (BRDF) [16, 30]. In real-time engines, computing direct illumination requires integrating the BRDF-light product. Dedicated techniques have been developed to integrate the GGX BRDF against different kinds of lights (probes, area lights, volumes, etc.). In this paper, we focus on *Linearly Transformed Cosines* (LTCs), which have been widely adopted as a means to integrate the GGX BRDF against area lights of various shapes [11–13]. For instance, LTCs are used in the Unity and Unreal engines for this purpose [3, 32]. However, their support is currently limited to the isotropic GGX BRDF, and thus anisotropic materials such as brushed metals cannot be shaded under area lighting with LTCs (see Figure 1). The objective of this work is to alleviate this limitation and bring real-time area lighting to the anisotropic GGX BRDF via LTCs.

More specifically, LTCs are spherical distributions with analytic integration properties over specific spherical domains. Thanks to the LTC approximation of GGX, the integral of the BRDF over the spherical domain covered by an area light can be computed analytically in real time (Fig. 2). LTCs are represented by 3×3 matrices M fitted to isotropic GGX lobes and stored in a small 2D look-up table [11]. Computing a similar look-up table for anisotropic GGX raises new challenges, which is the focus of this work.

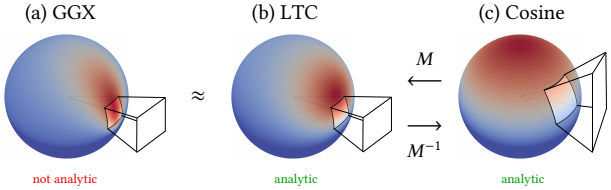


Figure 2: ***(a) A GGX lobe cannot be analytically integrated over the spherical domain covered by the area light. (b) An LTC represented by a matrix M provides a good approximation to the GGX lobe, and the integral equals (c) the analytic integral of a cosine lobe over the light transformed by M^{-1}.***

I3D'22, May 2022,

Objective. The crux of the problem is to obtain the 3×3 matrix M of the LTC that best approximates a given GGX lobe. Previously, Heitz et al. [11] proposed a fitting approach to compute a 2D look-up table

$$M = T_{\text{isoGGX}}(\theta, \alpha) \tag{1}$$

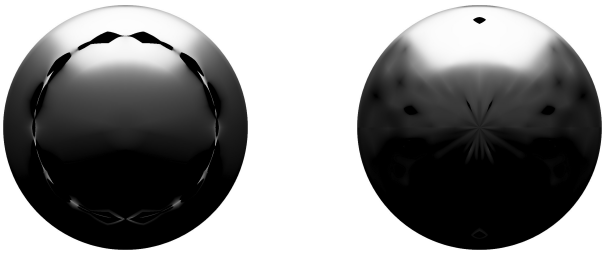
that approximates GGX lobes defined by the incidence angle θ and a roughness coefficient α. This is sufficient to cover the full isotropic GGX BRDF. Our objective is to compute a similar 4D look-up table

$$M = T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y) \tag{2}$$

that takes an additional azimuthal angle ϕ and anisotropic roughness coefficients (α_x, α_y). Note that once the matrix M is obtained, all of the existing applications of LTCs (area-light integration with various shapes, importance sampling, etc.) can be used without further modification. Thus, the only problem to solve is the precomputation of the 4D look-up table T_{anisoGGX}.

Contributions. The difficulty is that the fitting approach employed by Heitz et al. to compute T_{isoGGX} cannot simply be extended to compute T_{anisoGGX}. Indeed, in the isotropic case, the full dimensionality of LTCs is not used and this avoids several problems that arise in the anisotropic case. The artifacts highlighted in Figure 3 show that successfully bringing LTCs to anisotropic GGX requires *robust fitting* (Sec. 4), *well-defined interpolation* (Sec. 5), *valid symmetries* (Sec. 6) and *accurate storage* (Sec. 7 and 8). We introduce new mathematical properties of LTCs, such as non-uniqueness and axial symmetries, that are required to understand and overcome these failure cases. The final outcome of our method is a 4D look-up table that yields a plausible and artifact-free LTC approximation to anisotropic GGX and is small enough to be used in real time. We validate this table in the context of area-light shading with anisotropic GGX materials.

(a) broken fitted entry (Sec. 4) (b) ill-defined interpolation (Sec. 5)



(c) broken symmetry (Sec. 6) (d) early inversion (Sec. 7)

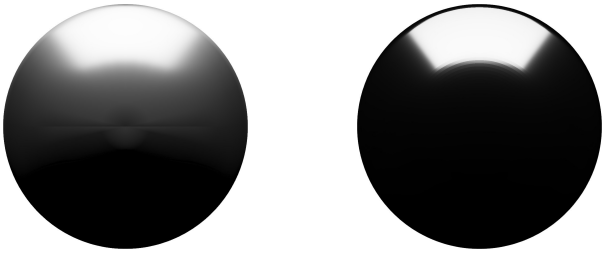


Figure 3: Illustration of the problems to overcome.

2 RELATED WORK

Real-time stochastic techniques. Recent graphics hardware makes it possible to use purely stochastic techniques such as reservoir sampling [4]. With stochastic approaches, integrating area lighting with arbitrary materials is simple but leads to noisy results. In the context of this paper, we instead aim to provide an *analytic shading* technique that produces a clean (noise-free) image.

Real-time analytic shading techniques. Even though stochastic techniques are appealing for the future, analytic methods remain important for today's real-time graphics. The first analytic solution to area lighting dates back to Lambert, who derived the irradiance from a polygonal light [20]. This early formula was brought to graphics for radiosity by Baum et al. [2] and was later extended by Arvo [1], who derived the integral over spherical polygons of cosines of arbitrary integer exponents (i.e., *Phong distributions*). Despite the detailed implementation of this technique provided by Snyder [29], it had limited practical impact due to the algorithmic complexity of the integration. In practice, real-time methods involved cheap approximations, mainly based on punctual evaluations [7, 19, 31]. As GPUs became more powerful, more accurate techniques arose, such as the approach of Lecocq et al., which was the first method to provide an accurate approximation for physically based materials while still being fast enough for real-time rendering [21, 22]. This method was later outperformed by *Linearly Transformed Cosines* (LTCs) [11], which remain today's leading approach for real-time area-light shading. We build on the state-of-the-art LTC method by adding support of anisotropic materials.

Applications of Linearly Transformed Cosines (LTCs). While LTCs were initially proposed for real-time polygonal-light shading, they have since found uses in many applications that will benefit from our contribution. The LTC analytic integration has been extended to other types of light, such as line lights [12] and sphere/disk lights [13]. Another important addition to LTC integration is shadowing. Though LTCs do not include visibility in the analytic shading integral, a low-variance ratio estimator has been proposed to incorporate shadows on top of the analytic shading integral [14]. An alternative approach consists of incorporating visibility by removing the edges of occluders in the LTC integral [18, 33]. The analytic integration property of LTCs has also proven useful in offline rendering, in the context of path guiding [6]. Besides integration, LTCs can also be importance sampled to provide noise-free ray tracing with very low samples per pixel [26]. Furthermore, LTCs have also found uses in differentiable rendering. Specifically, they have been used to select points on edges for efficient differentiable rendering [23] and to analytically compute gradients of the rendering equation [33]. Note that all of the aforementioned applications leverage properties of LTC distributions and work independently of how these distributions were fitted to a given material, and most of them use the isotropic GGX look-up table originally provided by Heitz et al. [11]. We provide a look-up table for anisotropic GGX.

BRDF fitting. There is a significant amount of work on the problem of fitting parametric models to BRDFs [8, 24] but the problem we address is different. BRDF fitting means fitting a 4D function with a simpler one. In our case, we fit the 2D outgoing-radiance lobe of the BRDF in each view-roughness configuration separately.

3 BACKGROUND

Here we review the mathematical background related to GGX and LTCs used in the subsequent sections. Note that the implementation of our method only requires Algorithms 1 and 2, with the rest used for plots, reference comparisons and technical proofs.

3.1 Background on the GGX BRDF

The GGX ("*Ground Glass Unknown*") microfacet BRDF was introduced by Walter et al. [30] and its anisotropic extension by Heitz [9]. The equations of this model are as follows:

Normalized directions. ω_v denotes the view direction and ω_l the light direction, with the following parameterizations:

$$\omega = (x, y, z) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta). \tag{3}$$

Normal Distribution Function (NDF). The NDF represents the statistical distribution of specular microfacets that reflect the incident light. It is parameterized by two roughness parameters (α_x, α_y):

$$D_{\text{ggx}}(\omega) = \frac{1}{\pi\,\alpha_x\,\alpha_y\left(\frac{x^2}{\alpha_x^2}+\frac{y^2}{\alpha_y^2}+z^2\right)^2}. \tag{4}$$

Masking-shadowing. The masking-shadowing function G_2 computes the attenuation due to the microsurface's self-shadowing:

$$G_2(\omega_v,\omega_l) = \frac{1}{1+\Lambda(\omega_v)+\Lambda(\omega_l)} \text{ with } \Lambda(\omega) = \frac{-1+\sqrt{1+\frac{\alpha_x^2\,x^2+\alpha_y^2\,y^2}{z^2}}}{2}. \tag{5}$$

Bidirectional Reflectance Distribution Function (BRDF). The cosine-weighted GGX BRDF is defined as

$$\rho(\omega_v,\omega_l)\,\cos\theta_l = \frac{F(\omega_v,\omega_h)\,D_{\text{ggx}}(\omega_h)\,G_2(\omega_v,\omega_l)}{4\,\cos\theta_v}, \tag{6}$$

where $\omega_h = \frac{\omega_v+\omega_l}{\|\omega_v+\omega_l\|}$ is the half vector and F is a Fresnel term. In the following, we do as Hill et al. [15] and always consider $F = 1$, since it can be reintroduced after the LTC approximation via a separate table. Equation (6) is used in the fitting approach of Heitz et al. [11]. We use this formula to make reference comparisons, but not in the implementation of our method.

Sampling. Sampling the Visible Normals Distribution Function (VNDF) [10] produces an approximate sampling of the cosine-weighted BRDF. The remaining weight of the samples is $\frac{1+\Lambda(\omega_l)}{1+\Lambda(\omega_v)+\Lambda(\omega_l)}$. In Section 4, we use the rejection-sampling Algorithm 1 to produce samples from the density that are perfectly proportional to the GGX cosine-weighted BRDF.

ALGORITHM 1: Sampling the cosine-weighted GGX BRDF.

```
while true do
    sample ω_h from the GGX VNDF /* Heitz's procedure [10] */
    ω_l = reflect(ω_v, ω_h)
    U ← rand() /* Uniform random number in [0, 1)          */
    if U < (1+Λ(ω_l)) / (1+Λ(ω_v)+Λ(ω_l)) then
        return ω_l
```

3.2 Background on LTCs

We now review properties of Linearly Transformed Cosines introduced by Heitz et al. [11] and illustrated in Figure 2-(b, c):

Definition. An LTC is defined as a matrix M that maps a clamped cosine distribution D_o to a spherical distribution defined as

$$D(\omega) = D_o(\omega_o)\frac{\partial\omega_o}{\partial\omega} = D_o\left(\frac{M^{-1}\omega}{||M^{-1}\omega||}\right)\frac{|M^{-1}|}{||M^{-1}\omega||^3}. \tag{7}$$

This equation is used by Heitz et al. [11] in their fitting procedure to precompute the look-up table of Equation (1). We use this formula to make a proof in Section 5, but not in the implementation of our method.

Area-light integration. The integral of an LTC D over the spherical domain $\mathcal{A}$ covered by an area light is the integral of the clamped cosine distribution D_o over the spherical domain $\mathcal{A}_o$ covered by the area light linearly transformed by M^{-1}:

$$\int_{\mathcal{A}} D(\omega)\,\mathrm{d}\omega = \int_{\mathcal{A}_o} D_o(\omega_o)\,\mathrm{d}\omega_o. \tag{8}$$

Like Heitz et al., we use this property at run time in the fragment shader to evaluate the integral of an LTC over an area light. The integration procedure depends on the shape of the light [11–13]. Note that our contribution relates to how the matrix M is obtained, which is independent of how the integration is computed.

Sampling. An LTC can be sampled by generating samples ω_o from the clamped cosine distribution and transforming them with the LTC matrix M, as shown in Algorithm 2. We use this algorithm in our fitting procedure, which is introduced in Section 4.

ALGORITHM 2: Sampling an LTC.

```
sample ω_o from a clamped cosine
ω = M ω_o / ||M ω_o||
return ω
```

4 FITTING

In this section, we address the problem of fitting an LTC represented by a matrix M to a GGX lobe, as shown in Figure 4.

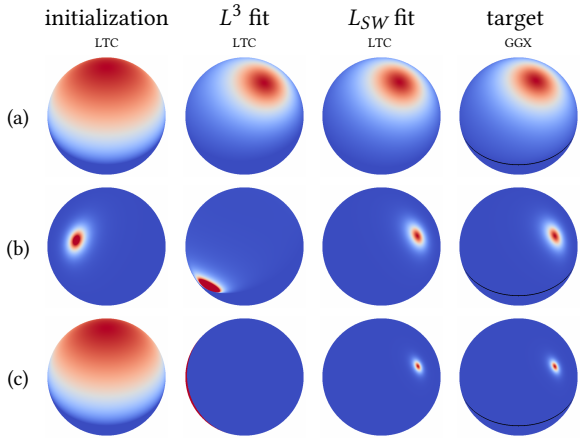


Figure 4: Fitting an LTC to a GGX lobe. ***We compare the*** L^3 ***fit of Heitz et al. to the*** L_{SW} ***fit we propose.***

4.1 Experimenting with the Previous Approach

Heitz et al. minimize the point-wise L_3 error between the cosine-weighted GGX BRDF (Eq. (6)) and the LTC (Eq. (7)). Their approach works in simple cases such as in Figure 4-(a), but we observed two main issues that make it unstable in more challenging configurations, leading to broken fits in our look-up table (Fig. 3-(a)).

Problem 1: null gradients. In Figure 4-(b), the LTC provided as a starting point does not overlap with the target distribution. The gradients of the L^3 error metric are thus close to 0 and the optimizer diverges.

Problem 2: high values. To avoid problem 1, in Figure 4-(c) we use a diffuse LTC (represented by an identity matrix M) as the starting point, such that there is significant overlap with the target GGX distribution. However, the target is sharp and evaluates to high values at the center of its lobe. These high values produce extremely high L^3 error gradients, which cause the optimizer to overshoot and stay trapped in a divergent configuration with null gradients (back to problem 1).

Discussion. Despite these issues, the L_3 optimization of Heitz et al. is successful because of the *accuracy of their starting points*. They use a diffuse LTC (an identity matrix M) for high roughnesses and initialize the matrix parameters with the already-optimized neighboring entries of the look-up table as the roughness decreases. The resolution of their table (64×64) ensures neighboring entries are close enough to provide accurate starting points. However, we need to aggressively reduce the resolution to store a 4D table (Sec. 8), so neighboring entries do not always overlap, especially with sharp distributions (low roughness). This is why we need an optimization process that is robust even with poor initialization.

4.2 Our Approach

Our objective is to find an optimization metric that is not subject to vanishing gradients or numerical instabilities with sharp distributions and works regardless of the accuracy of the initialization.

The Sliced Wasserstein loss. We use the *Sliced Wasserstein (SW)* loss [5, 27] between the direction samples of the target GGX lobe and the samples of the LTC distribution. This sample-wise loss approximates the optimal transport between two distributions and has shown several benefits in the machine learning community. The advantage over point-wise losses such as L_3 used by Heitz et al. is that it always provides smooth and stable gradients [17].

Definition. Consider two Probability Density Functions (PDFs) f and g and their respective marginals f_ω and g_ω over a random direction $\omega \in \Omega$. The SW distance between f and g is the expected difference between their respective Inverse Cumulative Distribution Functions (iCDF) F_ω^{-1} and G_ω^{-1} over all random directions:

$$L_{SW}(f,g) = \mathop{\mathbb{E}}_{\omega\in\Omega}\left[\int_0^1 \left|F_\omega^{-1}(u) - G_\omega^{-1}(u)\right| \,\mathrm{d}u\right]. \tag{9}$$

Despite this formulation appearing complicated at first sight, its implementation is extremely simple, as we shall see in Algorithm 3.

Discretization. An inverse CDF can be approximated by a list of sorted samples from the corresponding density. Hence, if we consider two sets of random samples $(f_1, ..., f_n)$ and $(g_1, ..., g_n)$ from respectively f and g and their *sorted* projections $(f_{1,\omega}, ..., f_{n,\omega})$ and $(g_{1,\omega}, ..., g_{n,\omega})$ onto direction ω, Equation (9) can be written as

$$L_{SW}(f,g) = \lim_{n\to\infty} \mathbb{E}_{\omega\in\Omega}\left[\frac{1}{n}\sum_{i=1}^{n}\left|f_{i,\omega} - g_{i,\omega}\right|\right], \tag{10}$$

i.e., the average of the differences between the sorted projections over the set of projection directions.

Optimization. In Algorithm 3, we compute a stochastic estimator of Equation (10) by sampling n random samples from the densities, projecting the samples onto a random direction ω, sorting the projections and averaging the absolute differences. Finally, we propagate the gradient of the loss back to the matrix M and do a gradient descent step. The variables highlighted in blue are the ones through which the gradients are propagated from L_{SW} back to M. Figure 5 illustrates the calculations. It is because this algorithm computes a mapping between samples rather than a difference between the densities that it is numerically stable even with sharp or non-overlapping densities, as shown in Figure 4. With this algorithm, we successfully fit all the entries of our look-up table T_{anisoGGX}.

ALGORITHM 3: Optimization step over the LTC matrix M with the Sliced-Wasserstein distance (blue variables depend on M).

Input: LTC matrix M
Input: GGX lobe view direction ω_v and roughnesses (α_x, α_y)
generate n random samples $(g_1, ..., g_n)$ from the GGX lobe
  /* Algorithm 1 */
generate n random samples $(f_1, ..., f_n)$ from the LTC distribution
  /* Algorithm 2 */
generate a random direction ω
$(f_{1,\omega}, ..., f_{n,\omega}) = \text{sort}(f_1 \cdot \omega, ..., f_n \cdot \omega)$
$(g_{1,\omega}, ..., g_{n,\omega}) = \text{sort}(g_1 \cdot \omega, ..., g_n \cdot \omega)$
$L_{SW} = \frac{1}{n}\sum_{i=1}^{n}\left|f_{i,\omega} - g_{i,\omega}\right|$
backpropagate gradient from L_{SW} to M
update $M = M - \epsilon \frac{\nabla L_{SW}}{\nabla M}$

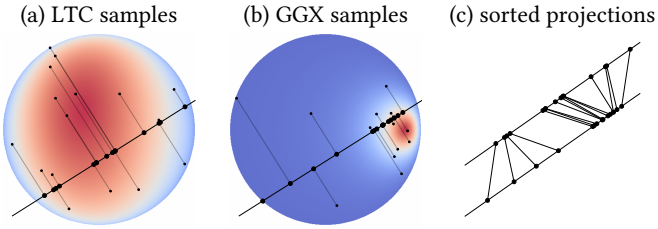


Figure 5: Illustration of Algorithm 3. ***We project random samples from the LTC (a) and the GGX lobe (b) onto a random direction and average the absolute differences between the sorted projections (c).***

Implementation. We implement this stochastic estimator in a differentiable calculus library, PyTorch [25], which provides automatic gradient backpropagation, and use the *Stochastic Gradient Descent (SGD)* [28] optimizer for M. We compute 10000 gradient descent steps and for each step we use $n = 2048$ random samples and average the estimator over 64 random directions.

5 INTERPOLATION

Even with the robust fitting approach described in the previous section, we noticed that our rendered results still suffered from jiggling artifacts shown in Figure 3-(b) when interpolating the entries of our fitted look-up table T_{anisoGGX}. In this section, we explain that the *non-uniqueness* of LTCs (5.1) makes interpolation ill-defined, and we propose a solution to this problem (5.2).

5.1 Non-Uniqueness of LTCs

The essential point of this section is that different matrices M can produce the same LTC distribution.

Intuitive explanation. An LTC is a cosine distribution that has undergone a linear transformation M (and a normalization). But the cosine distribution can also be *invariant* under some linear transformations: rotations R_z aligned with the z-axis and flipping matrices F_{xy} that flip the x- and/or y-axis. Hence, these linear operations can be appended to the matrix M without changing the resulting LTC distribution, as shown in Figure 6.

Property. For any rotation and flipping matrices

$$R_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad F_{xy} = \begin{bmatrix} \pm 1 & 0 & 0 \\ 0 & \pm 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

the LTC distributions associated with matrices M and $M\,R_z\,F_{xy}$ are the same.

Proof. We show that the evaluation of Equation (7) remains the same if we replace M by $M\,R_z\,F_{xy}$. First, the value of the Jacobian $\frac{\partial\omega_o}{\partial\omega}$ is unchanged because rotations or flips are area-preserving transformations (i.e., their Jacobians are 1). Second, the variable ω_o is mapped to another location where D_o (the cosine distribution) evaluates to the same value, since rotations around z, or x, y-axis flipping do not change the value of the cosine distribution.

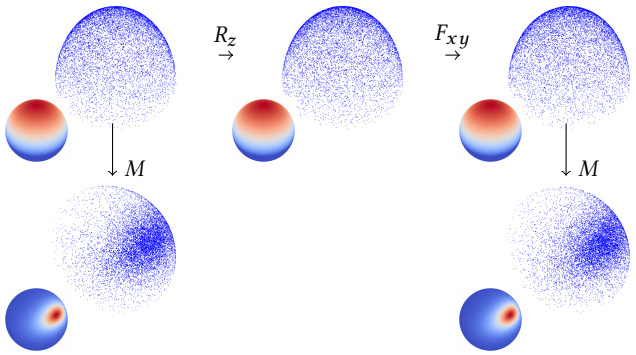


Figure 6: Non-uniqueness of LTCs. ***Cosine distributions remain unchanged under z-axis rotation*** R_z ***and*** xy ***flipping*** F_{xy}***. Linearly transformed cosines inherit this invariance.***

5.2 Well-Defined Interpolation with Alignment

The obvious way to interpolate LTCs consists of interpolating their matrices M. The non-uniqueness of the matrix M of a given LTC therefore has a direct impact on the interpolation behavior.

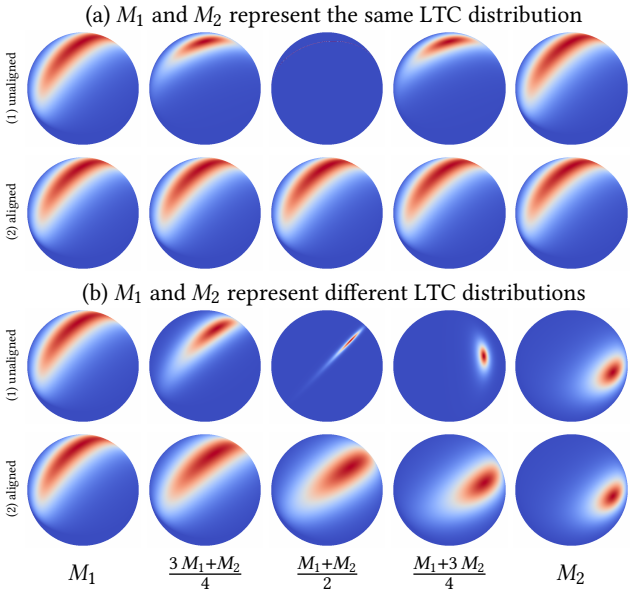


Figure 7: Interpolating LTC matrices with(out) alignment.

Naive interpolation. In Figure 7-(a1), we show that two different matrices that represent the same LTC distribution yield another unexpected distribution under interpolation. In the general case of Figure 7-(b1), where M_1 and M_2 represent different LTC distributions, interpolating the matrices does not produce a smooth transition between their respective distributions. Intuitively, this is because the matrices are *misaligned* due to the degrees of freedom introduced by their arbitrary rotation or flipping. This is what causes the interpolation artifacts shown in Figure 3-(b).

Aligned LTCs. Our idea is to *align* the LTC matrices M to cancel out rotation and flipping of the transformed cosine samples by aligning them with the original cosine samples. To do that, for a given M, we find the LTC matrix M_{aligned} that minimizes the average squared distance between the original cosine samples and their transformed counterparts, i.e., we optimize

$$M_{\text{aligned}} = \min_{F_{xy}} \min_{R_z} \mathbb{E}_{\omega_o \sim D_o} \left[\left\| \frac{M\,R_z\,F_{xy}\cdot\omega_o}{\|M\,R_z\,F_{xy}\cdot\omega_o\|} - \omega_o \right\|^2 \right]. \tag{12}$$

Intuitively, this optimization minimizes the average squared lengths of the lines in Figure 8. We implement it as a linear search over the rotation angle α and the four flipping cases.

Figure 8: LTC alignment. ***We minimize the average squared distance between the cosine (yellow) and LTC (green) samples.***

Robust interpolation. By aligning the LTC matrices before interpolating them, we obtain robust interpolation behavior. As expected, interpolating between the same distribution leaves it unchanged (Fig. 7-(a2)) and interpolating between different distributions produces smooth transitions (Fig. 7-(b2)). Aligning the entries of our fitted look-up table T_{anisoGGX} removes the interpolation artifacts of Figure 3-(b).

Discussion. Heitz et al. [11] do not report interpolation problems despite using a naive interpolation without alignment. This is because they only need four parameters of the LTC matrix to fit isotropic GGX: $M = \begin{bmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{bmatrix}$. A side effect of this special case is that the null entries of the matrix force its alignment and thus make the interpolation well-defined. The interpolation problem that we solve arises in the general case where all nine parameters of the LTC matrix are used, which is necessary for fitting anisotropic GGX. This is why we are, to our knowledge, the first to face the problem of LTC interpolation misbehaving, and to investigate the non-uniqueness property of LTCs.

6 SYMMETRIES

In this section, we leverage symmetries of the anisotropic GGX BRDF to remove numerical errors and reduce the storage induced by the dimensionality of our 4D fitted look-up table $T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y)$.

6.1 Parameterization of the Look-Up Table

Azimuthal symmetry. The GGX BRDF has axial symmetries over the x and y axes with respect to the view vector, as shown in Figure 9. We leverage this property to fit our look-up table over $\phi \in [0, \frac{\pi}{2}]$ rather than $\phi \in [0, 2\pi]$. We recover the full range $[0, 2\pi]$ in the following manner:

$$M = \begin{cases} \begin{bmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\phi) & \text{if } 0 \le \phi < \frac{\pi}{2}, \\ \begin{bmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\pi - \phi) & \text{if } \frac{\pi}{2} \le \phi < \pi, \\ \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\phi - \pi) & \text{if } \pi \le \phi < \frac{3\pi}{2}, \\ \begin{bmatrix} +1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(2\pi - \phi) & \text{if } \frac{3\pi}{2} \le \phi < 2\pi. \end{cases} \tag{13}$$

This reduces the size of the look-up table by a factor of four for the same angular resolution.

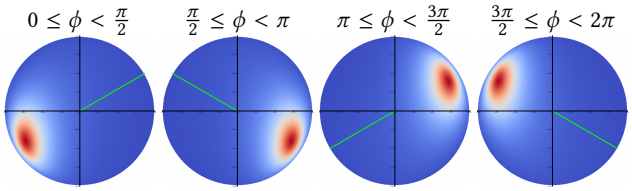


Figure 9: Azimuthal symmetries of the GGX BRDF. ***When the view vector (green) is symmetrized over the x and y axes, the GGX lobe undergoes the same symmetry.***

Roughness symmetry. The GGX BRDF has roughness symmetries shown in Figure 10 that can be written in the following manner:

$$T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \alpha_y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \frac{\pi}{2} - \phi, \alpha_y, \alpha_x). \tag{14}$$

Storing the data directly with the (α_x, α_y) parameterization virtually duplicates the data, so instead we use an alternative parameterization $T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$, where $\alpha \in [0, 1]$ is the largest roughness and $\lambda \in [0, 1]$ is the roughness ratio:

$$M = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \phi, \alpha_x, \frac{\alpha_y}{\alpha_x}) & \text{if } \alpha_x \ge \alpha_y, \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot T_{\text{anisoGGX}}(\theta, \frac{\pi}{2} - \phi, \alpha_y, \frac{\alpha_x}{\alpha_y}) & \text{otherwise.} \end{cases} \tag{15}$$

This increases the roughness resolution by a factor of two for the same storage.

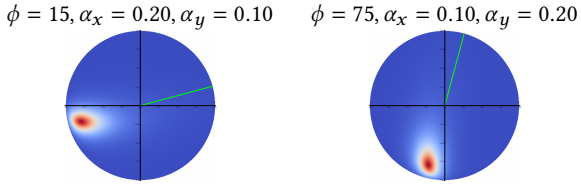


Figure 10: Roughness symmetries of the GGX BRDF. ***Permuting the x and y coordinates of the view vector and the roughnesses α_x and α_y produces the same permutation in the lobe's shape.***

6.2 Fixing Residual Errors in the Look-Up Table

Other symmetries of the GGX BRDF imply that certain entries of our fitted look-up table should be null. This would be the case if the fitting and alignment optimizations were perfect, but even small residual errors are sufficient to break these symmetries and produce the discontinuity and singularity artifacts present in Figure 3-(c). We can eliminate these artifacts by enforcing the expected symmetries in the entries of the look-up table:

Axial symmetries. The GGX lobe has axial symmetry over the x and y axes when, respectively, $\phi = 0$ and $\phi = \frac{\pi}{2}$ (Fig. 11-(a, b)).

Rotational symmetry. In upward views, where $\theta = 0$, the azimuthal angle ϕ does not contribute to the shape of the GGX lobe. Furthermore, the GGX lobe is centered on 0 and symmetric over the axes x and y (Fig. 11-(c, d)).

Look-up table post-processing. We post-process our look-up table to set the following entries (blue) to zero, and ensure that when $\theta = 0$ all of the entries match for the different values of ϕ:

$$T_{\text{anisoGGX}}(\theta, \phi) = \begin{cases} \begin{bmatrix} m_{00} & 0 & m_{02} \\ 0 & m_{11} & 0 \\ m_{20} & 0 & m_{22} \end{bmatrix} & \text{for } \phi = 0, \\ \begin{bmatrix} m_{00} & 0 & 0 \\ 0 & m_{11} & m_{12} \\ 0 & m_{21} & m_{22} \end{bmatrix} & \text{for } \phi = \frac{\pi}{2}, \\ \begin{bmatrix} m_{00} & 0 & 0 \\ 0 & m_{11} & 0 \\ 0 & 0 & m_{22} \end{bmatrix} & \text{for } \theta = 0 \text{ and } \phi = 0, \\ T_{\text{anisoGGX}}(\theta, 0), & \text{for } \theta = 0. \end{cases} \tag{16}$$

This post-processing step fixes the artifacts present in Figure 3-(c).

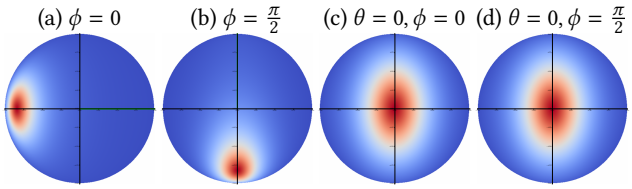


Figure 11: Symmetries of the GGX lobes with axis-aligned view directions.

7 MATRIX INVERSION

The LTC integration shown in Figure 2 actually uses the inverse matrix M^{-1}. Heitz et al. spare the inversion by storing and interpolating the inverse in their look-up table. However, interpolating the inverse creates severe distortions with our coarse resolution. This is because the diagonal coefficients of M that control the aperture of the LTC behave linearly with respect to α_x and α_y at low roughness while the diagonal coefficients of M^{-1} correlate with $\frac{1}{\alpha_x}$ and $\frac{1}{\alpha_y}$. We can therefore use a more aggressive discretization by storing and interpolating M instead of M^{-1}, and performing the inversion at run time in the fragment shader. To bring all of the entries of the look-up table into the same precision range, we divide M by the length of its third column, i.e., we constrain $M \cdot [0, 0, 1]$ to be a unit-length vector.

8 DISCRETIZATION

Heitz et al. [11] use a 64^2 resolution for their 2D look-up table $T_{\text{isoGGX}}(\theta, \alpha)$ with five floats per entry (they only use five parameters of the matrix M), which represents about 80 KB. If we were to use the same angular and roughness resolution, i.e., a 64^4 4D look-up table with nine floats per entry (we use the full matrix M), the memory requirement would be 576 MB. Thanks to the design choices introduced in the previous sections, we are able to reduce a resolution to 8^4. In Figure 12, we compare a resolution of 8^4 and 64^4 with respect to the GGX reference. The results show that the additional discretization error at 8^4 is negligible compared to the LTC approximation. As such, there is little to be gained by using a larger resolution.

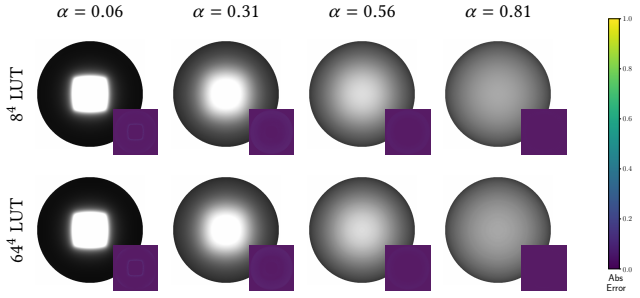


Figure 12: Renderings using 8^4 and 64^4 look-up tables. Inset difference images are with respect to the GGX reference.

9 IMPLEMENTATION OF OUR METHOD

In this section, we explain the implementation of our method, and summarize its requirements in Table 1.

Precomputations. We compute an 8^4 table $T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$ with a uniform discretization over $\theta \in [0, \frac{\pi}{2}]$, $\phi \in [0, \frac{\pi}{2}]$, $\alpha \in [0, 1]$ and $\lambda \in [0, 1]$. We compute each entry in the following way:

- We fit M to the corresponding anisotropic GGX lobe of roughnesses $\alpha_x = \alpha$ and $\alpha_y = \lambda\,\alpha$ using Algorithm 3 from Section 4.
- We align M by minimizing Equation (12) from Section 5, to ensure that interpolation is well-defined.
- We fix residual errors in M by applying Equation (16) from Section 6.
- We make M well-conditioned by dividing it by the length of its last column, as explained in Section 7.

The whole precomputation procedure is implemented in PyTorch and takes around two hours on an NVIDIA GeForce 2080 RTX GPU.

Storage. We store the parameters in 3D textures of resolution $64 \times 8 \times 8$ to benefit from trilinear hardware interpolation.

Run time. In the fragment shader, we proceed as follows:

- We get θ, ϕ, α_x, and α_y.
- We map ϕ to the first quarter $[0, \frac{\pi}{2}]$ using Equation (13) and compute α and λ using Equation (15).
- We fetch the 3D textures accordingly, manually interpolate over the last dimension, and apply the flipping and/or rotation matrices following Equations (13) and (15). This provides us with the matrix M.
- We invert M to obtain M^{-1}.

The cost of these operations is 0.610 ms for a full-screen quad at 1080p resolution with an NVIDIA GeForce RTX 2080 GPU, which is about four times the cost of the isotropic version.

LTC integration. Once M^{-1} is obtained, we use the existing LTC integration algorithms as in previous work [11–13]. Note that the overhead of our method only impacts the obtainment of M^{-1}, which can be amortized over the integration of multiple area lights.

Fresnel. We inject the Fresnel term of the GGX BRDF in the same way as Hill et al. [15]. We preintegrate the first and second Fresnel terms for each $(\theta, \phi, \alpha, \lambda)$ entry and store them as two additional channels in the look-up table that we interpolate at run time.

	Heitz et al. [11, 15]	ours
param.	$M^{-1} = T_{\text{isoGGX}}(\theta, \alpha)$	$M = T_{\text{anisoGGX}}(\theta, \phi, \alpha, \lambda)$
resolution	64×64	$8 \times 8 \times 8 \times 8$
containers	2D textures (64×64)	3D textures ($64 \times 8 \times 8$)
channels	5 (for M) + 2 (for Fresnel)	9 (for M) + 2 (for Fresnel)
memory	112 KB	176 KB
interpolation	HW 2D	HW 3D + SW 1D
inversion	-	fragment shader
total timing	0.160 ms	0.610 ms
LTC integration	0.110 ms/light	

Table 1: Requirements of our method.

10 RESULTS

In this section, we discuss the results produced by our method. Note that our supplemental material covers a dense set of plot and rendering configurations.

Plots. Figure 13 shows the fitted GGX lobes and our LTC approximation. We found out that the main limiting factor of our approximation is not our fitting technique but rather the representation power of LTCs. When the shape of the GGX lobe can be closely approximated by an LTC, our fitting technique is always successful (top rows in Fig. 13). However, the GGX lobe can exhibit *lune shapes* in certain configurations (high anisotropy, grazing view angle) and these shapes cannot be represented by LTCs (bottom rows in Fig. 13). Indeed, an LTC is a diffuse distribution transformed by a linear transformation. This allows for various first-order transformations such as changing the isotropic span of the lobe, elliptic anisotropy or skewness, but excludes lune-shaped lobes. In other words, it is not possible to accurately approximate these GGX configurations with LTCs, regardless of the fitting technique.

Renderings. Figure 14 shows the GGX reference and our LTC approximation. As expected from the plots, the approximation might have large errors compared to the reference but it remains plausible, and we did not find configurations where the result is visually unacceptable. Therefore, we believe that our approximation is good enough to be considered for non-predictive real-time rendering, but would discourage its use for more demanding applications.

11 CONCLUSION

We have proposed a method to bring Linearly Transformed Cosines to anisotropic GGX. It is the product of the experience we gained from many failed attempts. New insights into the mathematical properties of LTCs, careful design choices, and attention to detail were all crucial elements in ensuring a clean and artifact-free approximation. We believe that the proposed method provides a plausible approximation that passes the quality bar for game engines. It has low memory overhead compared to the isotropic version already used by practitioners. The 4D texture fetch is more expensive than the isotropic version but remains competitive for a real-time technique and can be amortized over multiple area lights. Thus, we don't see any barrier to using our anisotropic extension for video games. Furthermore, the same methodology could be applicable to other anisotropic materials. To facilitate reproduction, we plan to release the PyTorch fitting code, the fitted table and a minimalistic OpenGL demo that shows how to use it.

The main limitation is that our approximation is not accurate enough for all applications. For instance, we do not advise using it for predictive rendering or as an importance sampling technique for anisotropic GGX.

A notable finding is that the limiting factor of our approximation is the representation power of LTCs, which cannot produce all of the possible GGX shapes, such as lunes. Therefore, we believe our method reaches the limit of what is possible with LTC approximations of GGX BRDFs. Further improvements should thus be sought with a fundamentally different approach.

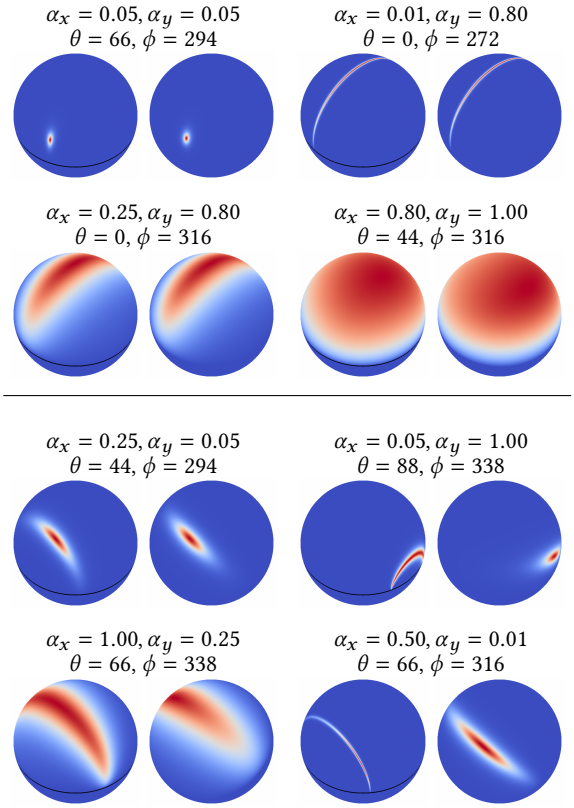


Figure 13: Plot results. ***We show the GGX reference (left) and our LTC approximation (right). More results in our supplemental material.***

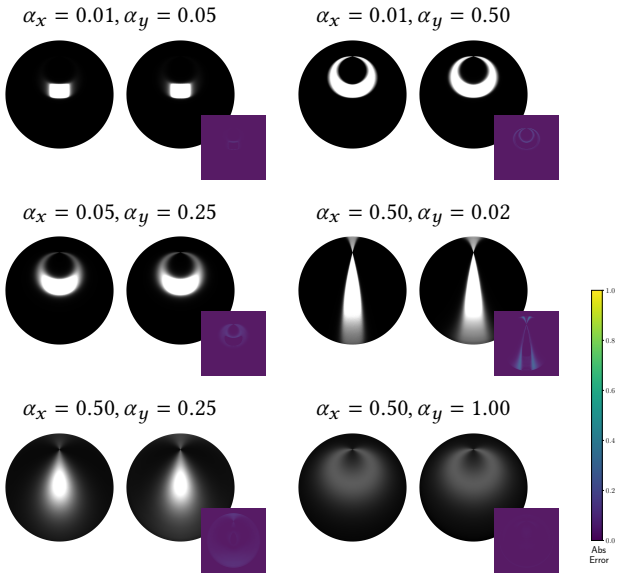


Figure 14: Rendered spheres with anisotropic materials and rectangular lights. ***We show the GGX reference (left), our LTC approximation (right), and the difference image. More results in our supplemental material.***

REFERENCES

[1] James Arvo. 1995. Applications of Irradiance Tensors to the Simulation of Non-Lambertian Phenomena. In *Proc. ACM SIGGRAPH*. 335–342.

[2] D. R. Baum, H. E. Rushmeier, and J. M. Winget. 1989. Improving Radiosity Solutions Through the Use of Analytically Determined Form-factors. *Computer Graphics (Proc. SIGGRAPH)* 23, 3 (1989), 325–334.

[3] Anis Benyoub. 2019. Leveraging Ray Tracing Hardware Acceleration In Unity. In *ACM SIGGRAPH Courses 2019*.

[4] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4 (2020).

[5] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and Radon Wasserstein Barycenters of Measures. *J. Math. Imaging Vis.* 51, 1 (2015), 22–45.

[6] Stavros Diolatzis, Adrien Gruson, Wenzel Jakob, Derek Nowrouzezahrai, and George Drettakis. 2020. Practical Product Path Guiding Using Linearly Transformed Cosines. *Computer Graphics Forum* (2020).

[7] MichałDrobot. 2014. Physically based area lights. In *GPU Pro 5*. 67–100.

[8] Jonathan Dupuy and Wenzel Jakob. 2018. An Adaptive Parameterization for Efficient Material Acquisition and Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (2018), 274:1–274:18.

[9] Eric Heitz. 2014. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques (JCGT)* 3, 2 (2014), 48–107.

[10] Eric Heitz. 2018. Sampling the GGX Distribution of Visible Normals. *Journal of Computer Graphics Techniques (JCGT)* 7, 4 (2018), 1–13.

[11] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-Time Polygonal-Light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.* 35, 4, Article 41 (2016).

[12] Eric Heitz and Stephen Hill. 2017. Linear-Light Shading with Linearly Transformed Cosines. In *GPU Zen*.

[13] Eric Heitz and Stephen Hill. 2017. Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines. In *ACM SIGGRAPH Courses 2017*.

[14] Eric Heitz, Stephen Hill, and Morgan McGuire. 2018. Combining Analytic Direct Illumination and Stochastic Shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. Article 2.

[15] Stephen Hill and Eric Heitz. 2016. Real-Time Area Lighting: a Journey from Research to Production. In *ACM SIGGRAPH Courses 2016*.

[16] Stephen Hill, Stephen McAuley, Laurent Belcour, Will Earl, Niklas Harrysson, Sébastien Hillaire, Naty Hoffman, Lee Kerley, Jasmin Patry, Rob Pieké, Igor Skliar, Jonathan Stone, Pascal Barla, Mégane Bati, and Iliyan Georgiev. 2020. Physically Based Shading in Theory and Practice. In *ACM SIGGRAPH 2020 Courses*. Article 11, 12 pages.

[17] Soheil Kolouri, Gustavo K. Rohde, and Heiko Hoffmann. 2018. Sliced Wasserstein Distance for Learning Gaussian Mixture Models. In *Conference on Computer Vision and Pattern Recognition, CVPR 2018*.

[18] Aakash Kt, Parikshit Sakurikar, and P. J. Narayanan. 2021. Fast Analytic Soft Shadows from Area Lights. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.).

[19] Sébastien Lagarde and Charles de Rousiers. 2014. Physically Based Shading in Theory and Practice: Moving Frostbite to PBR. In *ACM SIGGRAPH Courses 2014*.

[20] Johann Heinrich Lambert. 1760. Photometria, sive De mensura et gradibus luminus, colorum et umbrae. (1760).

[21] Pascal Lecocq, Arthur Dufay, Gaël Sourimant, and Jean-Eudes Marvie. 2016. Accurate analytic approximations for real-time specular area lighting. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 113–120.

[22] Pascal Lecocq, Gaël Sourimant, and Jean-Eudes Marvie. 2015. Accurate Analytic Approximations for Real-time Specular Area Lighting. In *ACM SIGGRAPH 2015 Talks*. Article 68, 1 pages.

[23] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (2018).

[24] Addy Ngan, Frédo Durand, and Wojciech Matusik. 2005. Experimental Analysis of BRDF Models. In *Proceedings of the Eurographics Symposium on Rendering*. 117–226.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.

[26] Christoph Peters. 2021. BRDF Importance Sampling for Polygonal Lights. *ACM Trans. Graph.* 40, 4 (2021), 14 pages.

[27] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2012. Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision*. 435–446.

[28] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[29] John M. Snyder. 1996. *Area Light Sources for Real-Time Graphics*. Technical Report MSR-TR-96-11. Microsoft Research.

[30] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. In *Proceedings of the Eurographics Symposium on Rendering Techniques 2007*. 195–206.

[31] Lifeng Wang, Zhouchen Lin, Wenle Wang, and Kai Fu. 2008. *One-Shot Approximate Local Shading*. Technical Report.

[32] Marcus Wassmer, Jerome Platteaux, Arne Schober, and Ignacio Llamas. 2018. Cinematic Lighting in Unreal Engine. In *Game Developers Conference 2018*.

[33] Yang Zhou, Lifan Wu, Ravi Ramamoorthi, and Ling-Qi Yan. 2021. Vectorization for Fast, Analytic, and Differentiable Visibility. *ACM Trans. Graph.* 40, 3 (2021).