

# Real-Time Polygonal-Light Shading with Linearly Transformed Cosines

Eric Heitz  
Unity Technologies

Jonathan Dupuy  
Unity Technologies

Stephen Hill  
Ubisoft

David Neubelt  
Ready At Dawn Studios



**Figure 1:** We use *Linearly Transformed Cosines* to shade physically based materials with polygonal lights in real-time. Our technique supports arbitrary, non-convex, potentially textured polygonal lights. The shading time for the primary lighting pass in each of these configurations is 2.4ms, 7.5ms, and 2.4ms, respectively. Timings are for an image resolution of  $1920 \times 1080$  pixels, for the area-light shader only, without MSAA, with an NVIDIA GeForce GTX 980 GPU.

## Abstract

In this paper, we show that applying a linear transformation—represented by a  $3 \times 3$  matrix—to the direction vectors of a spherical distribution yields another spherical distribution, for which we derive a closed-form expression. With this idea, we can use any spherical distribution as a base shape to create a new family of spherical distributions with parametric roughness, elliptic anisotropy and skewness. If the original distribution has an analytic expression, normalization, integration over spherical polygons, and importance sampling, then these properties are inherited by the linearly transformed distributions.

By choosing a clamped cosine for the original distribution we obtain a family of distributions, which we call *Linearly Transformed Cosines* (LTCs), that provide a good approximation to physically based BRDFs and that can be analytically integrated over arbitrary spherical polygons. We show how to use these properties in a real-time polygonal-light shading application. Our technique is robust, fast, accurate and simple to implement.

**Keywords:** Shading, BRDF, area lighting, real-time rendering

**Concepts:** •Computing methodologies → Reflectance modeling;

## 1 Introduction

Physically based shading involves the computation of the illumination equation, i.e. integrating the product of a *Bidirectional Reflectance Distribution Function* (BRDF) and the lighting in the spherical domain. In this paper, we are primarily interested in shading from polygonal lights, which implies computing the integration of a BRDF over a spherical polygon. Despite polygonal lights being theoretically one of the simplest lighting models, they are challenging in real-time rendering for two main reasons:

- Integrating parametric spherical distributions over spherical polygons is difficult in general, even with the simplest distributions. For instance, the analytic Phong-polygon integration

is prohibitively costly [Arvo 1995] and there is no analytic solution for Spherical Gaussians [Xu et al. 2014].

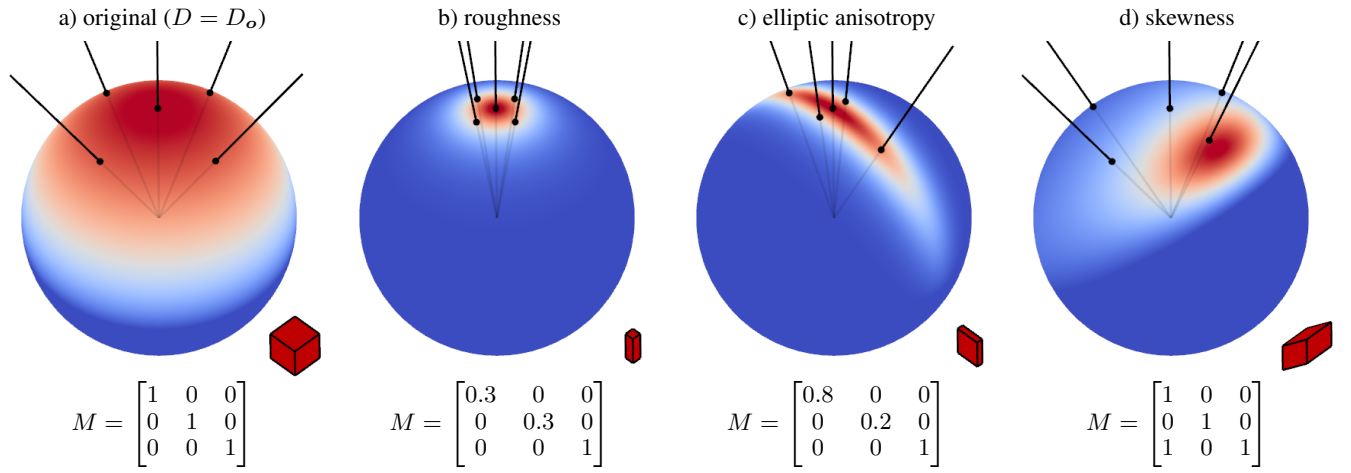
- State-of-the-art physically based material models are not simple distributions [Hill et al. 2015]; they have sophisticated shapes with anisotropic stretching and skewness that need to be represented for the material to appear realistic.

In Sec. 3, we introduce *Linearly Transformed Spherical Distributions* (LTSDs), a new kind of spherical distribution that solves these two problems. We start from an original spherical distribution and apply a linear transformation—represented by a  $3 \times 3$  matrix—to its direction vectors. This yields a parameterization that allows us to modify the shape of the original distribution, such as roughness, elliptic anisotropy and skewness, as illustrated in Fig. 2. Thanks to this parameterization, we can use any spherical distribution to create a new family of parametric spherical distributions with different base shapes, as shown in Fig. 3. The main feature of these distributions is that they inherit several properties of the original distribution such as normalization, integration over arbitrary spherical polygons, and importance sampling.

In Sec. 4, we show that using a clamped cosine for the original distribution yields a family of distributions—which we call *Linearly Transformed Cosines* (LTCs)—that provide a good approximation to physically based BRDFs thanks to the wide variety of spherical shapes they cover. Furthermore, since a clamped cosine distribution can be analytically integrated over arbitrary spherical polygons, so can the LTCs. In Sec. 5, we show how to use this property in a real-time polygonal-light shading application.

## 2 Previous Work

**Polygonal-Light Shading** Analytic solutions to polygonal lighting are currently limited to cosine-like distributions. Integrating a clamped cosine over a polygonal domain—i.e. computing the irradiance from a polygon—was solved centuries ago by Lambert [1760] and introduced to computer graphics by Baum et al. [1989]. This was extended by Arvo [1995] to Phong distributions, i.e. cosines with an exponent that controls the sharpness of the distribution. A detailed implementation of Arvo’s technique is provided by Snyder [1996]. The main limitation of this approach is that the complexity of the Phong-polygon integration is  $O(e \cdot n)$ , where  $n$  is the number of polygon vertices and  $e$  the Phong exponent, i.e. the cost of the integration increases with the sharpness of the distribution. This is why video-games use cheaper approx-



**Figure 2:** Parameterizing spherical distributions with linear transformations. We generate new distributions by applying a linear transformation, represented by matrix  $M$ , to the direction vectors associated with the original distribution. Its effect can be seen on the lines and the red cube. In this figure  $D_o$  is a clamped cosine distribution (a). The matrix  $M$  provides control over roughness (b), anisotropy (c) and skewness (d) of the transformed distribution.

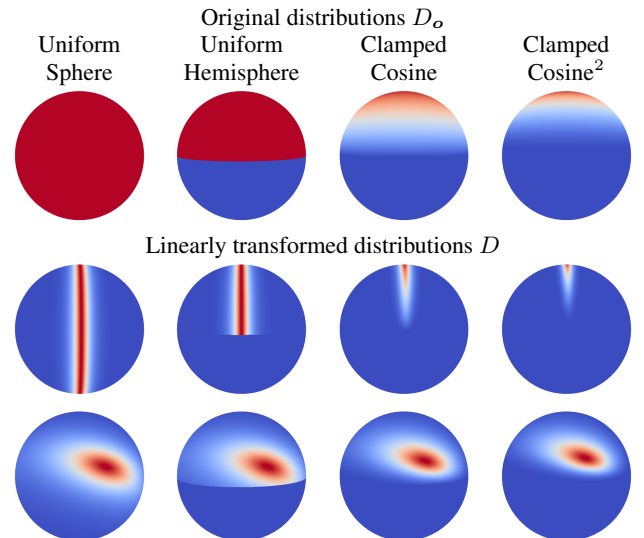
imations instead, such as the *most-representative-point* heuristic, where the area light is replaced with a point light [Wang et al. 2008; Drobot 2014]. While such techniques are cheap enough to be used in real-time, they are inaccurate and do not guarantee robust integration: the result can be off by several orders of magnitude and exhibit visual artifacts in some configurations [Lagarde and de Rousiers 2014]. The most recent concurrent work dedicated to polygonal lights is the technique of Lecocq et al. [2015]. They provide an approximate solution to the Phong-polygon integration in  $O(n)$ , i.e. independent of the sharpness of the distribution. Their integration is fast enough to be computed in real-time. However, it remains approximate and limited to the rotationally symmetric shapes of Phong lobes. They propose to use the Phong distribution as a microfacet distribution, which needs to be integrated over a non-polygonal halfspace domain. They approximate this domain with a polygon and compute their approximate Phong integration over it. Although this approach recovers the anisotropy of microfacet models, the successive approximations result in visible artifacts in some configurations. While the complexity of the integration with our technique is also  $O(n)$ , the result is exact and our distribution covers complex spherical shapes with elliptic anisotropy and skewness. Our technique is also much simpler to implement: it boils down to multiplying the  $n$  polygon’s vertices by a matrix before computing its irradiance with the classic Lambert formula.

**Spherical Distributions** There are few spherical distributions that represent sophisticated shapes and at the same time provide useful integration properties. Spherical Harmonics are limited to low-frequency shapes. The most common all-frequency distributions used in computer graphics are rotationally symmetric lobes: either Phong distributions [Phong 1975] or Spherical Gaussians, also known as von Mises-Fisher distributions [Fisher 1953]. Despite their simplicity, integrating them is already a challenging problem. As discussed above, approximations are required for efficient integration of Phong distributions over spherical polygons; similarly, integrating Spherical Gaussians over sharp spherical domains can only be done approximately and requires precomputed look-up tables [Iwasaki et al. 2012; Xu et al. 2014]. Furthermore, designing more sophisticated distributions often means giving up on simple properties. For instance, distributions with elliptic anisotropy from the field of statistics [Bingham 1974; Kent 1982] have no useful analytic operators. Anisotropic Spherical Gaussians (ASGs) were designed to provide approximate operators for such distributions in computer graphics applications [Xu et al. 2013].

However, computing their normalization factor—their integral over the sphere—already requires the evaluation of a complex formula. Deriving a practical solution for integration over spherical polygons therefore seems unlikely.

### 3 Linearly Transformed Spherical Distributions

In this section, we introduce Linearly Transformed Spherical Distributions (LTSDs). We show how an original distribution can be reshaped by applying a linear transformation to its direction vectors (Fig. 2), we define the new distributions obtained in this way (Fig. 3), and we discuss their properties.



**Figure 3:** Linearly Transformed Spherical Distributions. The choice of the original distribution allows us to create families of parametric distributions with different base shapes.

### 3.1 Definition

**Original Distribution to be Transformed**  $D_o$  denotes the original distribution that we reparameterize with linear transformations. The choice of  $D_o$  controls the base shape of the transformed distributions, as shown in Fig. 3.

**Linear Transformations** To generate a new distribution  $D$ , we apply a linear transformation represented by a  $3 \times 3$  matrix  $M$  to the direction vectors  $\omega_o$  of the original distribution  $D_o$ . Fig. 2 shows how the choice of  $M$  affects the properties of the distribution. Note that throughout the paper, we assume normalized direction vectors, i.e. the exact transformation<sup>1</sup> is  $\omega = M \omega_o / \|M \omega_o\|$  and, reciprocally, we recover the original direction with the inverse transformation:  $\omega_o = M^{-1} \omega / \|M^{-1} \omega\|$ .

**Closed-Form Expression** The magnitude of an LTSD is the magnitude of the original distribution  $D_o$  in the original direction  $\omega_o$  multiplied by the change of solid angle measure due to the distortion of the spherical transformation. It has the closed-form expression:

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

where  $\frac{\partial \omega_o}{\partial \omega} = \frac{|M^{-1}|}{\|M^{-1} \omega\|^3}$  is the Jacobian of the transformation of normalized direction vectors  $\omega = M \omega_o / \|M \omega_o\|$ . We provide a proof of this result in Appendix A.

**Shape Invariance** The distribution is invariant to scaling transformations ( $M = \lambda I$ ) since scaling vectors doesn't change their directions. Furthermore, rotations only change the orientation of the distribution, not its shape. This is because the Jacobian of the spherical transformation is constant if  $M$  is a scaling or rotation matrix:  $\frac{\partial \omega_o}{\partial \omega} = 1$ .

**Median Vector** With several classic original distributions  $D_o$ , the  $z$ -vector  $(0, 0, 1)^T$  is the median vector, i.e. any plane that contains this vector cuts  $D_o$  into two parts of equal weight  $\frac{1}{2}$ . This property is preserved by the transformation: the transformed median vector is the median vector of the transformed distribution  $D$ . In all of the configurations of Fig. 2, the right column of  $M$  is  $(0, 0, 1)^T$ , i.e. the median vector remains aligned with the  $z$ -axis. For instance, in Fig. 2.d, the skewness squashes the distribution on one side and stretches it on the other side, and the  $z$ -axis remains the median of the distribution.

### 3.2 Properties

The transformed distribution  $D$  inherits several properties of the original distribution  $D_o$ .

**Normalization** The norm of an LTSD  $D$  is the norm of its original distribution  $D_o$ . By using Eq. (1) we get

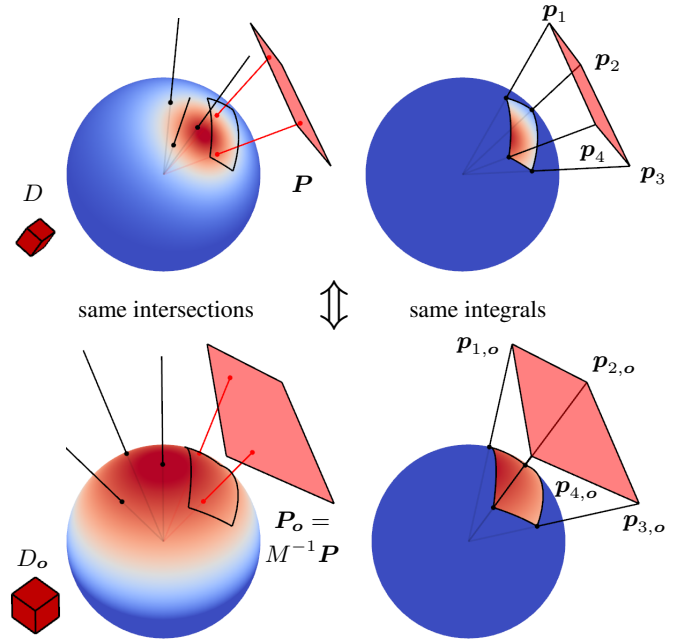
$$\int_{\Omega} D(\omega) d\omega = \int_{\Omega} D_o(\omega_o) \frac{\partial \omega_o}{\partial \omega} d\omega = \int_{\Omega} D_o(\omega_o) d\omega_o. \quad (2)$$

<sup>1</sup>Note that the normalization after the linear transformation makes the final transformation  $\omega = M \omega_o / \|M \omega_o\|$  non-linear. We have chosen to keep the adjective “linear” because it provides the intuition of how the original distribution is transformed by  $M$ . Furthermore, some properties of LTSDs, such as the invariance of the polygonal integration (Fig. 4), are directly due to the linearity of the 3D transformation before the normalization.

**Integration over Polygons** The integral of an LTSD  $D$  over a polygon  $P$  is the integral of the original distribution  $D_o$  over the polygon  $P_o = M^{-1} P$  obtained by applying the inverse transformation  $M^{-1}$  to each vertex of the polygon:

$$\int_P D(\omega) d\omega = \int_{P_o} D_o(\omega_o) d\omega_o, \quad (3)$$

which we obtain with the same change of variable as in Eq. (2). We provide an explanation of this result in Fig. 4. Intuitively, the integral in Eq. (3) is the probability that a direction sampled in  $D$  intersects the polygon  $P$ . Any linear transformation applied to both the direction vectors of  $D$  and the polygon  $P$  does not change these intersections, and hence preserves the value of the integral. We use this property by applying  $M^{-1}$  that transforms the distribution  $D$  back into the original distribution  $D_o$ , and the polygon into another polygon  $P_o = M^{-1} P$ . As a result, the integral of  $D$  over  $P$  equals the integral of  $D_o$  over  $P_o$ .



**Figure 4:** Invariance of the polygonal integration. The configuration on the bottom is the top configuration multiplied by matrix  $M^{-1}$ . Since the linear transformation doesn't change line-polygon intersections, samples generated in the distribution have the same probability of intersecting the polygon in both configurations. The integral of the distribution over the spherical polygon is thus the same in both configurations.

**Importance Sampling** Thanks to the definition of LTSDs, if  $D_o$  can be importance sampled then so can be  $D$ . As shown in Algorithm 1, we generate a random sample  $\omega_o$  from  $D_o$  and we transform  $\omega = \frac{M \omega_o}{\|M \omega_o\|}$ . Because the LTSD defined in Eq. (1) is  $D_o$  multiplied by the Jacobian of this transformation, the probability density function (PDF) of the sample  $\omega$  is exactly  $D(\omega)$ .

#### Algorithm 1 Importance Sampling

sample $\omega_o$ from $D_o$	$\triangleright \text{PDF}(\omega_o) = D_o(\omega_o)$
return $\omega = \frac{M \omega_o}{\ M \omega_o\ }$	$\triangleright \text{PDF}(\omega) = D_o(\omega_o) \frac{\partial \omega_o}{\partial \omega} = D(\omega)$

## 4 Approximating Physically Based BRDFs with Linearly Transformed Cosines

In this section, we show that choosing a clamped cosine distribution for the original distribution allows us to create a family of linearly transformed distributions that yield a good approximation to physically based BRDFs.

**Linearly Transformed Cosines** For the original distribution  $D_o$  we choose a normalized clamped cosine distribution in the hemisphere given by the  $z$  direction:

$$D_o(\omega_o = (x, y, z)) = \frac{1}{\pi} \max(0, z), \quad (4)$$

and Linearly Transformed Cosines (LTCs) are defined by substituting  $D_o$  in Eq. (1).

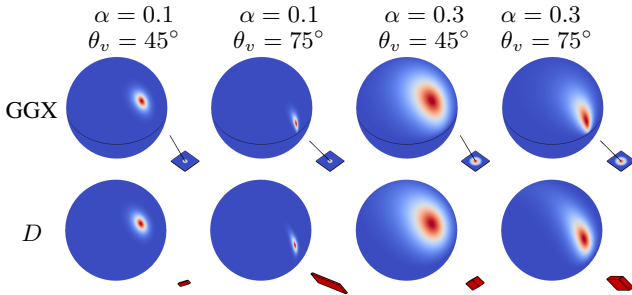
**Fitting** We choose to approximate the GGX microfacet BRDF [Walter et al. 2007], which is currently considered to be the most realistic parametric BRDF [Hill et al. 2015]. More specifically, we approximate the spherical functions given by the cosine-weighted BRDFs over the light directions  $\omega_l$ :

$$D \approx \rho(\omega_v, \omega_l) \cos \theta_l. \quad (5)$$

For an isotropic material, the BRDF depends on the incident direction  $\omega_v = (\sin \theta_v, 0, \cos \theta_v)$  and the roughness parameter  $\alpha$ . For any combination  $(\theta_v, \alpha)$  we fit the cosine-weighted BRDF with a Linearly Transformed Cosine, i.e. we find the matrix  $M$  that yields the best fit. Thanks to the planar symmetry of isotropic BRDFs, and since Linearly Transformed Cosines are scale invariant,  $M$  can be represented as

$$M = \begin{bmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{bmatrix}, \quad (6)$$

so only 4 parameters need to be fitted. Empirically, we found that minimizing the  $L^3$  error yields the best visual results in terms of shading. Fig. 5 shows results of our fitting.



**Figure 5:** Fitting parametric BRDFs with Linearly Transformed Cosines. We precompute a set of Linearly Transformed Cosines that provide the best fits for a GGX microfacet BRDF with varying incident angle  $\theta$  and roughness  $\alpha$ . Our distribution captures the anisotropy and skewness of the target distribution. More results in our supplemental material.

**Representation and Storage** The polygonal integral in Eq. (3) only needs the inverse matrix  $M^{-1}$ . After the fitting procedure, we store the inverse matrices in a precomputed table. Similar to Eq. (6),  $M^{-1}$  can be represented by 4 parameters for isotropic BRDFs. Furthermore, we use an additional parameter for the norm  $\int_{\Omega} \rho(\omega_v, \omega_l) \cos \theta_l d\omega_l$  of the fitted BRDF. We store a total of 5 parameters in 2D float textures of resolution  $64 \times 64$ , parameterized by  $\theta \in [0, \frac{\pi}{2}]$  and  $\sqrt{\alpha} \in [0, 1]$ , and fetched with linear interpolation. The size of the precomputed data is 80KB.

## 5 Real-Time Polygonal-Light Shading with Linearly Transformed Cosines

Shading with diffuse polygonal lights means computing the illumination integral over a polygonal domain:

$$I = \int_P L(\omega_l) \rho(\omega_v, \omega_l) \cos \theta_l d\omega_l, \quad (7)$$

where  $\omega_v$  is the view direction,  $\rho$  the BRDF,  $P$  the polygon, and  $L$  the radiance emitted by the polygon and received from direction  $\omega_l$ . In order to use the properties of Linearly Transformed Cosines, at runtime we fetch matrix  $M^{-1}$  of the Linearly Transformed Cosine  $D$  associated with incident angle  $\theta_v$  and roughness  $\alpha$  of the BRDF, and we use the approximation of Eq. (5) in Eq. (7):

$$I = \int_P L(\omega_l) \rho(\omega_v, \omega_l) \cos \theta_l d\omega_l \approx \int_P L(\omega_l) D(\omega_l) d\omega_l. \quad (8)$$

### 5.1 Shading with Constant Polygonal Lights

If the radiance emitted by the polygonal light is spatially constant, i.e.  $L(\omega_l) = L$ , it becomes a separable multiplication factor of the integral

$$I = \int_P L(\omega_l) D(\omega_l) d\omega_l = L \int_P D(\omega_l) d\omega_l. \quad (9)$$

Thanks to the polygonal integration property from Eq. (3), we simplify

$$\int_P D(\omega) d\omega = \int_{P_o} D_o(\omega_o) d\omega_o = E(P_o), \quad (10)$$

where  $E$  is the irradiance of the polygon  $P_o = M^{-1}P$ . Indeed, since  $D_o$  is a clamped cosine distribution, its integral over a polygon is the irradiance of this polygon, which has a closed-form expression [Baum et al. 1989]:

$$E(\mathbf{p}_1, \dots, \mathbf{p}_n) = \frac{1}{2\pi} \sum_{i=1}^n \text{acos}(\langle \mathbf{p}_i, \mathbf{p}_j \rangle) \left\langle \frac{\mathbf{p}_i \times \mathbf{p}_j}{\|\mathbf{p}_i \times \mathbf{p}_j\|}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle, \quad (11)$$

with  $j = (i + 1) \bmod n$ . Note that the formula assumes a spherical polygon located in the upper hemisphere. In practice, we clip each edge of the polygon before adding its contribution in the sum of Eq. (11).

### 5.2 Shading with Textured Polygonal Lights

In this section, we assume that the radiance  $L(\omega_l)$  emitted by the polygon is represented by a 2D color texture applied to it. In this case it cannot be separated from the integral. We rewrite the Eq. (8):

$$I \approx \int_P L(\omega_l) D(\omega_l) d\omega_l = I_D I_L, \quad (12)$$

$$I_D = \int_P D(\omega_l) d\omega_l, \quad (13)$$

$$I_L = \frac{\int_P L(\omega_l) D(\omega_l) d\omega_l}{\int_P D(\omega_l) d\omega_l}. \quad (14)$$

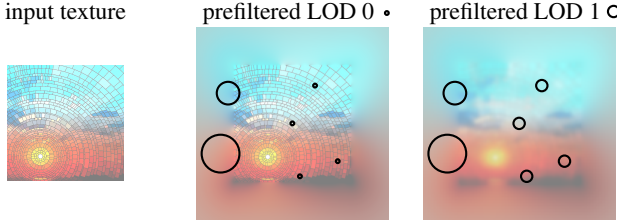
The advantage of this formulation is that it allows us to break the problem into two subproblems—computing  $I_D$  and  $I_L$ —that have different properties.

**The Shape of the Highlight** The value of  $I_D$  is the integral of the distribution over the polygonal domain, i.e. it is the fraction of rays in  $D$  that intersect  $P$ . It drives the order of magnitude of the integral  $I$  and shapes the highlight. We compute its exact value with Eq. (3).

**The Color of the Highlight** In contrast,  $I_L$  can be seen as the average color gathered by the rays in  $D$  that intersect  $P$  and is responsible for the color of the highlight. It can be formulated as a texture-space filter, which we propose to approximate with pre-filtered textures. In the following, we describe how to prefilter the texture with Levels of Detail (LODs) and how to parameterize a texture fetch.

### 5.3 Texture Prefiltering

Eq. (14) defines  $I_L$  as the values of the texture  $L$  passed through a filter  $F(\omega_l) = \frac{D(\omega_l)}{\int_P D(\omega_l) d\omega_l}$  projected in texture space. In the prefiltering step, we approximate  $F$  with a texture-space Gaussian filter. There are two properties of  $F$  to be preserved by the approximate filter:  $F$  is clamped to the polygon, i.e.  $F = 0$  outside the texture, and  $F$  is normalized inside the texture:  $\int_P F(\omega_l) d\omega_l = 1$ . In practice, it means that the approximate texture-space filter cannot leak outside the texture. To achieve this, we clamp and renormalize the texture-space Gaussian inside the texture. Furthermore, the values of the prefiltered texture have to be defined everywhere in texture space, even outside the texture. As illustrated in Fig. 6, we introduce a margin around the texture. In this region, we increase the radius of the filter so that it intersects the texture. This is necessary for the filter to be well-defined. Finally, beyond the margin, we clamp texture coordinates to the edge. This representation defines a smooth, low-frequency function everywhere in texture space outside the texture. It is a property expected from the exact filter  $F$ .



**Figure 6:** Texture Prefiltering. The circles show the radius of the texture-space filter at various locations. Inside the texture, the radius is chosen according to the LOD. Outside the texture, we increase the radius so that it intersects the texture.

### 5.4 Texture Fetching

#### On the Difficulty of Fetching Prefiltered Area-Light Textures

Parameterizing a texture fetch requires two quantities: the texture coordinates and the LOD. A naive approach consists of computing the coordinates, in texture space, of the intersection of the average (or median) direction of the distribution. This approach is unstable and sometimes ill-defined. For instance, in Fig. 7.a, the average direction of the distribution fails to intersect the texture plane. Another difficulty is that the appropriate LOD depends on several parameters: the sharpness of the distribution, and the inclination and the distance of the texture plane.

**Simplifying the Problem with the Cosine Configuration** To simplify the problem, we express it in the cosine configuration associated with  $D$ . This transformation is illustrated in Fig. 7. As for polygonal integration in Eq. (3), we apply the inverse matrix  $M^{-1}$

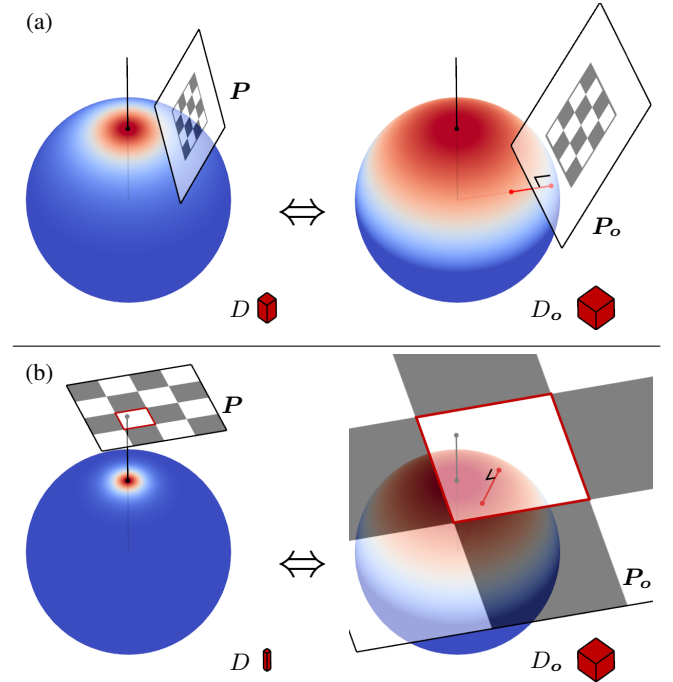
to the configuration, i.e. we transform

$$I_L = \frac{\int_P L(\omega_l) D(\omega_l) d\omega_l}{\int_P D(\omega_l) d\omega_l} = \frac{\int_{P_o} L_o(\omega_l) D_o(\omega_l) d\omega_l}{\int_{P_o} D_o(\omega_l) d\omega_l}, \quad (15)$$

where  $P_o = M^{-1}P$  is the polygon in the cosine configuration and  $L_o(\omega_l)$  the associated transformed texture. The advantage of this configuration is that parameterizing a texture-space filter that approximates the cosine distribution  $D_o$  is much simpler and robust than for an arbitrary distribution  $D$ .

#### Fetching the Prefiltered Texture in the Cosine Configuration

In the cosine configuration, we use the orthonormal projection of the shading point onto the texture plane (Fig. 7). This point is always well-defined, simple to compute, and yields accurate texture fetches regardless of the sharpness of distribution  $D$ , as shown in Fig. 7.b. Furthermore, since it is an orthonormal projection, the texture plane inclination at this point is exactly  $90^\circ$  and therefore doesn't introduce anisotropy to the texture-space filter shape. We neglect the low-frequency variations of the cosine distribution and the anisotropic distortion of the texture parameterization after the polygon's transformation. The choice of the LOD reduces to a 1D function of the ratio between the squared distance  $r^2$  to the texture plane and the area  $A$  of the polygon. We choose the LOD prefiltered with a Gaussian of standard deviation  $\sigma = \sqrt{\frac{r^2}{2A}}$  in texture space.



**Figure 7:** Texture fetching. The configuration on the right is the left configuration multiplied by matrix  $M^{-1}$ . (a) The average direction of the distribution (black) doesn't intersect the texture plane and cannot be used for fetching. In the cosine configuration, we use the orthonormal projection of the shading point onto the texture plane (red), which is always well-defined. (b) In the cosine configuration we fetch the texel expected from the sharp distribution. This is because the angular difference between the average direction (black) and the orthonormal projection used for the fetch (red) is compensated for by the stretched polygon.

## 6 Results

**Game Engine Integration** Our technique is easy to integrate into an existing real-time rendering pipeline. Fig. 8 shows a scene in which the lighting is computed with our technique in a video-game engine. The scene contains 8 rectangular lights and the total rendering time per frame is 15ms at  $1280 \times 720$  on an NVIDIA GeForce GTX 980 GPU.



**Figure 8:** Integration into a game engine. The scene contains 8 rectangular lights and the total rendering time per frame is 15ms on an NVIDIA GeForce GTX 980 GPU at  $1280 \times 720$  pixels.

**Performance** We benchmarked our shader by rasterizing a full-screen quad covering  $1280 \times 720$  pixels on an NVIDIA GeForce GTX 980 GPU. With the rectangle (4 edges) from Fig. 9, the shading time is 0.64ms and it is independent of the roughness  $\alpha$  of the approximated GGX BRDF. Furthermore, the performance doesn't change with the additional texture fetch. This is because the bottleneck of the shading is located in the polygon's irradiance computation. With the star-shaped polygon (10 edges) from Fig. 10, the shading time is 1.66ms. As expected from Eq. (3), the algorithm scales linearly with the number of polygon edges.

**Comparison Against Ground Truth** In Figs. 9, 10, 11, and 12 we show comparisons of our technique against ground-truth results. We compute the ground truth by importance sampling the GGX microfacet BRDF and we raytrace the area lights. Even though our technique introduces some error compared to the ground truth, it always yields visually plausible results, which are often accurate. The exact LTC-polygon integration also guarantees robustness and avoids special cases or singularities, unlike the *most-representative-point* heuristic [Lagarde and de Rousiers 2014]. To understand the sources of errors, recall that the core idea of Sec. 5.2 is to separate the computation of the highlight's shape and colors. The approximation that affects the shape of the highlight is the BRDF fitting. The results show that the fitting is less accurate for high roughness and grazing incidence. This is consistent with the fitting results

from Fig. 5. The approximation that affects the colors of the highlight is related to the prefiltered texture fetches. Fig. 11 shows that we fetch the expected texels with sharp, almost specular BRDFs, as explained in Sec. 5.4 and Fig. 7. The difference becomes noticeable with rough reflections, i.e. wide texture-space filters. With homogeneous or low-frequency textures, any sufficiently wide filter will return a value close to the average of the texture, so it doesn't need to be accurate to match the ground truth. However, the filter shape is important with heterogeneous, or highly-varying texture content, and our isotropic Gaussian filter is only a rough approximation of the texture-space projection of the clamped cosine. A noticeable difference can be seen in the stretching of the highlight's colors that is present in the ground truth but absent from our results. Whether parameterizing anisotropic texture fetches in our technique could significantly improve those results is worth further investigation. Fig. 12 shows failure cases where the textures are too heterogeneous for the approximate filter shape to be accurate. The errors result in a global bias in the first case and artifacts in the second case.

## 7 Conclusion and Future Work

We have derived *Linearly Transformed Spherical Distributions* (LTSDs), a new way of parameterizing spherical distributions that covers a variety of spherical shapes with several analytic properties, such as closed-form expressions, normalization, integration over polygons, and importance sampling. We applied this idea to polygonal shading with physically based BRDFs using *Linearly Transformed Cosines*, one instance of these new distributions.

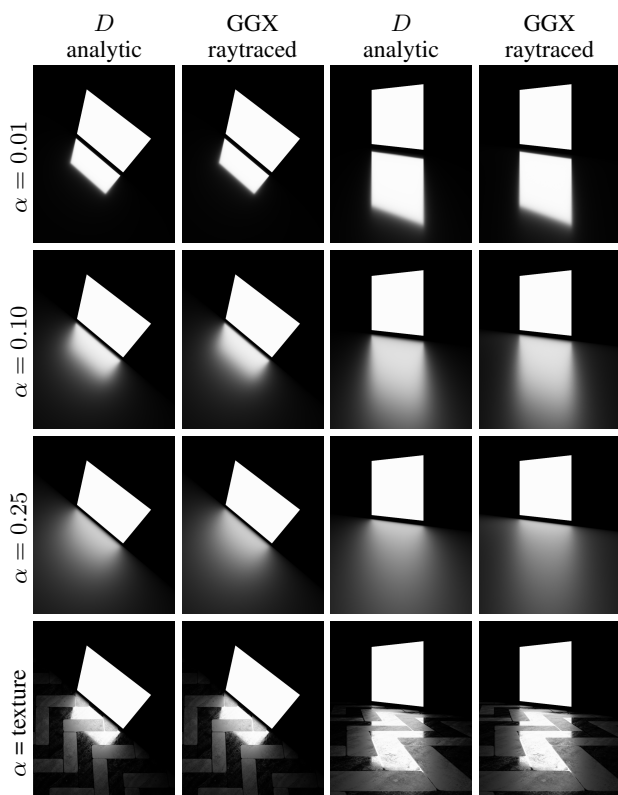
**Integration over Other 3D Shapes** It is worth noting that the integration invariance of LTSDs is not limited to polygons, but also applies to any class of 3D surfaces invariant to 3D linear transformations. For instance, linearly transformed ellipsoids remain ellipsoids. Hence, if an original distribution can be integrated over an ellipsoid then so can the linearly transformed distributions, with the same change of variable as in Eq. (3). For instance, if a solution were available for computing the irradiance from an ellipsoid, we could directly use it to integrate ellipsoids against Linearly Transformed Cosines. The same idea applies to ellipses.

**Deriving Other Operators** We currently do not know whether it is possible to derive other analytic operators that could be useful for rendering applications, such as the inner product or convolution.

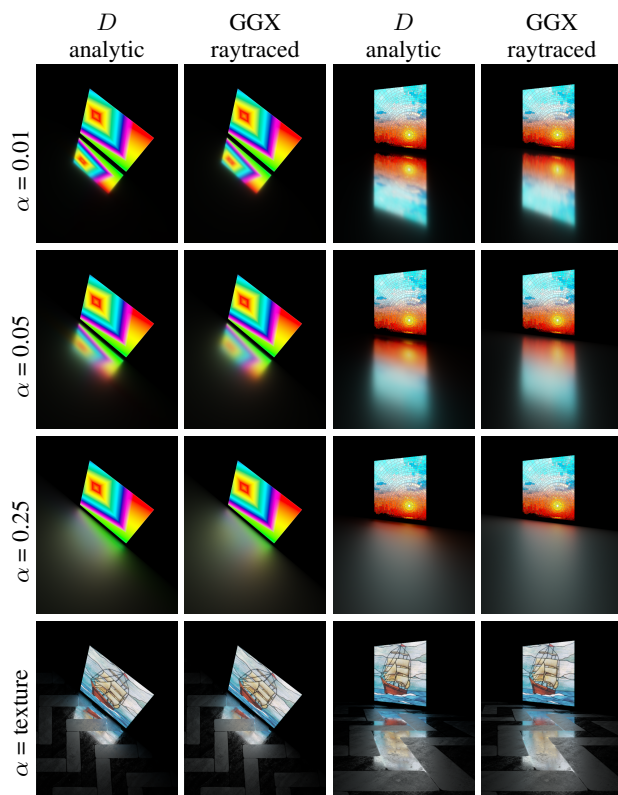
**Parameter Estimation** We would also like to investigate whether it is possible to fit LTSDs to a target distribution by directly extracting a matrix  $M$  instead of using non-linear optimization as we currently do for physically based BRDFs. For instance, we have shown that the median vector is preserved by linear transformations, and it directly yields the third column of matrix  $M$ . Finding other invariants of LTSDs are thus a possibility worth exploring for parameter estimation.

## Acknowledgements

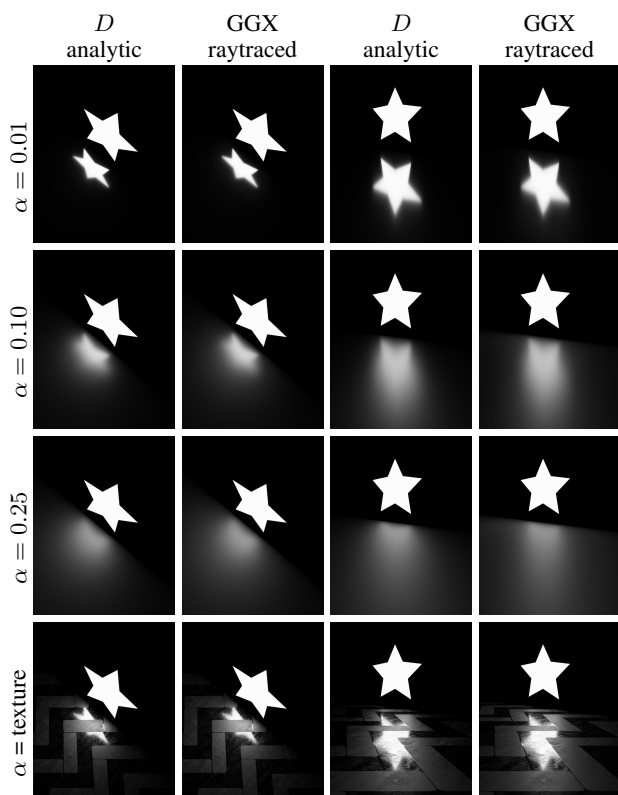
The authors wish to thank Robert Cupisz for integrating the polygonal-light shading technique into the Unity engine and the Unity Demo Team artists for creating the scene. The pictures in Figure 8 were rendered thanks to them. The authors also thank Branimir Karadžić for the bgfx graphics library and generous technical support. In addition, the authors acknowledge Frank Meinel and Morgan McGuire for the Sponza scene, as well as Alexandre Pestana for updated materials.



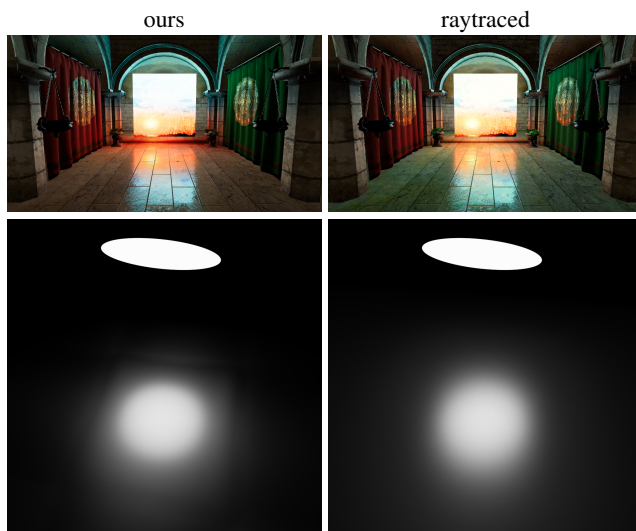
**Figure 9:** Comparison against ground truth with a constant polygonal light. *More results in our supplemental material.*



**Figure 11:** Comparison against ground truth with textured polygonal lights. *More results in our supplemental material.*



**Figure 10:** Comparison against ground truth with a constant, non-convex polygonal light. *More results in our supplemental material.*



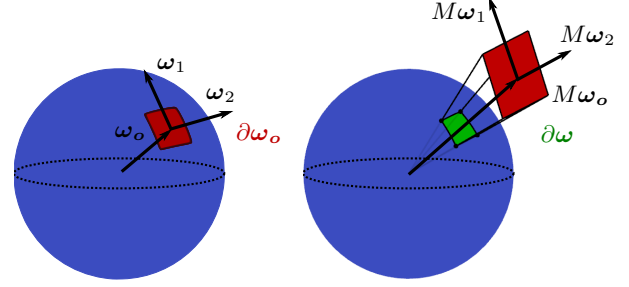
**Figure 12:** Failure cases of our texture fetching. We use our pre-filtered texture fetching technique with a stained glass texture (top) and a texture mask emulating a disk (bottom). In these cases, the textures are too heterogeneous for the approximate filter shape to be accurate. The errors result in a global bias in the first case (top) and artifacts in the second case.

## References

- ARVO, J. 1995. Applications of irradiance tensors to the simulation of non-lambertian phenomena. In *Proc. ACM SIGGRAPH*, 335–342.
- BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. 1989. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics (Proc. SIGGRAPH)* 23, 3, 325–334.
- BINGHAM, C. 1974. An antipodally symmetric distribution on the sphere. *The Annals of Statistics* 2, 6, 1201–1225.
- DROBOT, M. 2014. Physically based area lights. In *GPU Pro 5*, 67–100.
- FISHER, R. 1953. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 217, 1130, 295–305.
- HILL, S., MCAULEY, S., BURLEY, B., CHAN, D., FASCIONE, L., IWANICKI, M., HOFFMAN, N., JAKOB, W., NEUBELT, D., PESCE, A., AND PETTINEO, M. 2015. Physically based shading in theory and practice. In *ACM SIGGRAPH Courses 2015*.
- IWASAKI, K., FURUYA, W., DOBASHI, Y., AND NISHITA, T. 2012. Real-time rendering of dynamic scenes under all-frequency lighting using integral spherical gaussian. *Computer Graphics Forum (Proc. of Eurographics)* 31, 727–734.
- KENT, J. T. 1982. The Fisher-Bingham distribution on the sphere. *Journal of the Royal Statistical Society. Series B (Methodological)* 44, 1, 71–80.
- LAGARDE, S., AND DE ROUSIERS, C. 2014. Physically based shading in theory and practice: Moving Frostbite to PBR. In *ACM SIGGRAPH Courses 2014*.
- LAMBERT, J. H. 1760. *Photometria, sive de mensura et gradibus luminis, colorum et umbrae*.
- LECOCQ, P., SOURIMANT, G., AND MARVIE, J.-E. 2015. Accurate analytic approximations for real-time specular area lighting. In *ACM SIGGRAPH 2015 Talks*, 68:1–68:1.
- PHONG, B. T. 1975. Illumination for computer generated pictures. *Computer Graphics (Proc. SIGGRAPH)* 18, 6, 311–317.
- SNYDER, J. M. 1996. Area light sources for real-time graphics. Tech. Rep. MSR-TR-96-11, Microsoft Research.
- WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Proc. Eurographics Symposium on Rendering*, 195–206.
- WANG, L., LIN, Z., WANG, W., AND FU, K. 2008. One-shot approximate local shading. Tech. rep.
- XU, K., SUN, W.-L., DONG, Z., ZHAO, D.-Y., WU, R.-D., AND HU, S.-M. 2013. Anisotropic Spherical Gaussians. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 32, 6, 209:1–209:11.
- XU, K., CAO, Y.-P., MA, L.-Q., DONG, Z., WANG, R., AND HU, S.-M. 2014. A practical algorithm for rendering interreflections with all-frequency brdfs. *ACM Transactions on Graphics* 33, 1, 10:1–10:16.

## A Derivation of the Jacobian

We derive the Jacobian of Eq. (1). The Jacobian of the spherical transformation measures the change of solid angle illustrated in Fig. 13. We start from an original configuration with an orthonormal basis  $(\omega_o, \omega_1, \omega_2)$  and an infinitesimal solid angle  $\partial\omega_o$  represented by the red area. We transform the configuration by multiplying by matrix  $M$ . The new infinitesimal solid angle to be measured is  $\partial\omega$ , represented by the green area. It is the surface area of the sphere covered by the projection of the transformed infinitesimal surface

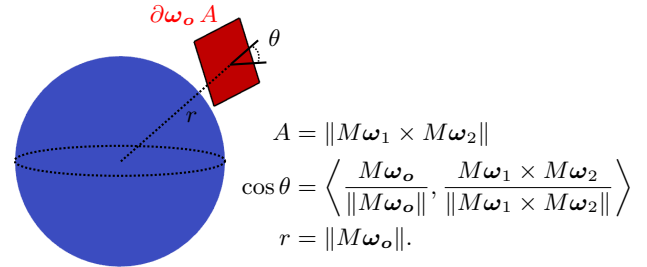


**Figure 13:** Jacobian of the spherical transformation. The Jacobian measures the scaling  $\frac{\partial\omega}{\partial\omega_o}$  of the solid angle from the transformation.

Fig. 14 illustrates how to compute the new solid angle  $\partial\omega$ . It is given by

$$\partial\omega = \partial\omega_o A \frac{\cos \theta}{r^2}, \quad (16)$$

where  $\partial\omega_o A$  is the area of the solid angle (or surface element) after the transformation,  $r$  its distance to the center of the sphere and  $\theta$  the angular difference between its direction and its normal.



**Figure 14:** Computation of the transformed solid angle.

To expand and simplify Eq. (16), we use the linear algebra property

$$\begin{aligned} M\omega_1 \times M\omega_2 &= |M|M^{-T}(\omega_1 \times \omega_2) \\ &= |M|M^{-T}\omega_o, \end{aligned} \quad (17)$$

and after simplifying we obtain

$$\frac{\partial\omega}{\partial\omega_o} = \frac{|M|}{\|M\omega_o\|^3}, \quad (18)$$

and reciprocally

$$\frac{\partial\omega_o}{\partial\omega} = \frac{|M^{-1}|}{\|M^{-1}\omega\|^3}, \quad (19)$$

which is the Jacobian used in Eq. (1).