BRDF Importance Sampling for Linear Lights

Christoph Peters

1 Karlsruhe Institute of Technology

Clipped solid angle sampling, 1.1 ms
Our projected solid angle sampling, 1.1 ms
Our projected solid angle and LTC sampling with MIS, 1.5 ms
Reference

11.285
8.863
0.583

Figure 1: The bistro exterior (2.9 million triangles), lit by a long linear light source. We compute shading using Monte Carlo integration with ray traced shadows. Taking one sample per pixel proportional to solid angle yields moderate noise throughout the scene. Our projected solid angle sampling achieves clean diffuse shading outside penumbrae but specular highlights remain noisy. If we take a second sample proportional to a linearly transformed cosine [HDHN16] and combine both techniques using clamped optimal MIS [Pet21], noise outside penumbrae becomes weak everywhere. Timings are full frame times at 1920 × 1080 on an NVIDIA RTX 2080 Ti, numbers are RMSEs.

Abstract

We introduce an efficient method to sample linear lights, i.e. infinitesimally thin cylinders, proportional to projected solid angle. Our method uses inverse function sampling with a specialized iterative procedure that converges to high accuracy in only two iterations. It also allows us to sample proportional to a linearly transformed cosine. By combining both sampling techniques through suitable multiple importance sampling heuristics and by using good stratification, we achieve unbiased diffuse and specular real-time shading with low variance outside penumbrae at two samples per pixel. Additionally, we provide a fast method for solid angle sampling.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Through new graphics hardware, real-time ray tracing has become practical. Nonetheless, real-time renderers have to limit their ray budget to a few rays per pixel. With generic path tracers, the variance at such low sample counts is too high. GPU-friendly importance sampling techniques for specific light transport phenomena are in high demand. We provide such a method for direct lighting with linear lights, which are an excellent idealization of fluorescent tubes.

According to the reflection equation, the reflected radiance in direction $\omega_o \in \Omega$ is

$$L_o(\omega_o) = \frac{1}{\Omega} \int L_i(\omega_i) V(\omega_i) f(\omega_i, \omega_o) \langle n, \omega_i \rangle \ d\omega_i$$

where $\Omega \subset \mathbb{R}^3$ is the hemisphere around the surface normal $n \in \mathbb{R}^3$, $f$ is the bidirectional reflectance distribution function (BRDF), $L_i$ gives incoming radiance due to the light source, $V(\omega_i) \in \{0, 1\}$ is light visibility and $\langle n, \omega_i \rangle$ denotes a dot product for the cosine term. A Monte Carlo estimator takes random samples $\omega_i$ from $\Omega$ proportional to a known density $p(\omega_i)$ and estimates the integral as

$$\frac{L_i(\omega_i)V(\omega_i)f(\omega_i, \omega_o) \langle n, \omega_i \rangle \ d\omega_i}{p(\omega_i)}.$$

To cancel most of the variance, $p(\omega_i)$ should be nearly proportional to the integrand.

For direct lighting with Lambertian emitters, $L_i(\omega_i)$ is constant within the solid angle of the light source and zero elsewhere. Then techniques that sample this solid angle uniformly...
An ideal sampling technique would distribute samples proportional to BRDF times cosine but only within the solid angle of the light source. With such a technique, the only remaining source of noise is the visibility term \( V(\omega_i \mid \omega_o) \). Sampling proportional to the visibility term requires global scene knowledge but neglecting it only introduces noise in penumbrae.

To provide such a technique for diffuse BRDFs, we sample linear lights proportional to the cosine term \( n \cdot \omega_o \). In other words, we sample their projected solid angle uniformly. Our method accomplishes this goal by inverse function sampling with a highly optimized numerical inversion (Sec. 3). Our error analysis shows that it always converges to high accuracy in only two iterations. The implementation is thoroughly optimized and some of these ideas also apply to solid angle sampling of linear lights.

Once we can sample linear lights proportional to a cosine, we can also sample them proportional to a linearly transformed cosine (LTC) [HDHN16]. That gives rise to a suitable sampling strategy for specular BRDFs (Sec. 4). We combine this strategy with projected solid angle sampling through clamp optimized MIS [Pet21].

Our sampling techniques enable unbiased shading for linear lights, with ray traced shadows and minimal noise outside of penumbrae (Fig. 1). Blue noise dithering and uniform jittered sampling [RAMN12] are highly effective for the one-dimensional linear lights (Sec. 5.3). Hence, even the shadows have good quality at two samples per pixel (one specular, one diffuse). Two prior works offer similar functionality but they are either prone to branch divergence [LADL18] or slower by a constant factor [Pet21] (Sec. 5.4).

The full source code of our renderer is available.

2. Related Work

There is extensive prior work on linear lights, partly because fluorescent tubes are widely used and partly because integration in one dimension is easier than in two. Early work on diffuse shading [NON85] solves special cases in closed form and relies on quadrature for the general case. Approximate [PA91] and exact [BP93] closed-form solutions for Phong shading followed. Pi-cott [Pic92] presents a closed form for diffuse shading using a slightly different formulation, where the linear light is not an infinitesimally thin cylinder but a sequence of point lights. For specular shading, he proposes a most representative point approach, an approximation that persists in real-time rendering until today [Dro14, dCI17].

These older works advocate variants of shadow volumes [NON85, PA91, BP93, Pic92] whereas our method offers sampling for Monte Carlo integration. Ramamoorthi et. al. [RAMN12] use ray tracing and study the impact of different strategies for stratification on shadows (see Sec. 5.3). As a form of importance sampling, Gamito [Gam16] samples the solid angle of cylinders and disks of finite radius uniformly. The method samples a bounding rectangle using an exact method [UnFK13] and rejects samples outside of the relevant solid angle at the caps. Our method does not reject any samples because it takes the cylinder radius to zero in the limit.

Solid angle sampling is available for all common types of area lights. For spheres [Wan92] and triangles [Arv95, Pet21], there are closed-form solutions. For ellipses and ellipsoids [Hei17] a method involving Newton-Raphson iterations and look-up tables exists [GUUnK∗17]. Still more effective importance sampling for diffuse shading takes samples proportional to the cosine term \( n \cdot \omega_o \) (like our technique). For spheres that can be done through iterative root finding [UnG18] or in closed form [PD19]. For polygons, there are methods based on recursive subdivision [Un00], Newton’s method [Arv01] or special iterative algorithms [Pet21]. All of these could be combined with rejection sampling [Gam16] to sample cylinders of finite size in proportion to projected solid angle.

However, we strive for a faster, more specialized solution. Projected solid angle sampling and LTC importance sampling of linear lights are also useful for differentiable rendering because moving edges make strong contributions to derivatives [LADL18]. The implementation used there employs costly Newton bisection for inverse function sampling. Our method uses a more specialized procedure that always converges in two iterations and is more thoroughly optimized.

Recently, sampling problems have been studied more fundamentally. The triangle cut parametrization warps samples of a suitable approximate density into samples of another density [Hei20]. In one dimension, it still consumes two random numbers such that stratification is lost. Hart et al. [HPM∗20] approximate a target density in primary sample space with linear or quadratic polynomials and sample proportional to those.

LTCs [HDHN16] provide good approximations of many specular BRDFs. With this approximation, computation of unshadowed specular shading reduces to computation of the projected solid angle of the transformed light source. We use them for sampling. They also work for linear lights [HH17a] and disk lights [HH17b]. Dupuy et al. [DBH17] present a similar method for specular importance sampling of spherical lights. It is efficient but struggles with anisotropic highlight shapes.

Moureau et al. [MPC19] describe a GPU-friendly hierarchical data structure to select important lights among thousands of dynamic lights. Most renderers combine strategies for sampling of light sources with methods to sample in proportion to the BRDF [Hd14] through multiple importance sampling (MIS) [VG95]. However, this approach is ineffective for linear lights since all samples miss the light.

3. Sampling Linear Lights for Diffuse Shading

Our method samples the solid angle of linear lights exactly proportional to the cosine term \( n \cdot \omega_o \). Then the Monte Carlo estimator for Lambertian emitters is proportional to \( V(\omega_i \mid f(\omega_o, \omega_i)) \). For
As starting point, we need a proper definition of a linear light and we need to compute its solid angle. These considerations are not novel [NON85, BP93] but serve to fix the notation and to establish building blocks for an efficient algorithm. We follow the notion (unlike Picott [Pic92]) that a linear light is an infinitesimally thin cylinder and its solid angle. Looking straight at a cap of the cylinder, we see that the solid angle has an opening angle \( \phi_R \) with respect to the shading point is

\[ \phi_R := \arcsin \frac{R}{||q||}. \]

In the \( st \)-plane, the cylinder appears as axis-aligned rectangle of height \( 2R \) (Figure 2c). Since we eventually take \( R \) to zero, there is no need to bother with the exact shape of the caps. We capture the extent by storing \( s \)-coordinates for the normalized end points of the line:

\[ l_{0,s} := \frac{(l_0,s)}{||l_0||} \in \mathbb{R}, \quad l_{1,s} := \frac{(l_1,s)}{||l_1||} \in \mathbb{R}. \]

Now we are prepared to compute the solid angle of the cylinder (with incorrect caps). We write it as integral over the hemisphere in cylindrical coordinates (see [PH16] chapter 13.6.1):

\[ \Omega_R := \int_{l_{0,s}}^{l_{1,s}} \int_{q_0}^{q_\pi} 1 \, d\phi \, d\omega_\phi = 2\phi_R (l_{1,s} - l_{0,s}). \]  

The angle \( \phi \) is an azimuthal coordinate around the central axis of the cylinder and \( \omega_\phi \in [-1,1] \) is the \( s \)-coordinate of a unit-direction vector. For \( R = 0 \), this solid angle is zero because \( \phi_R = 0 \). Thus, we divide out \( R > 0 \) before we take the limit:

\[ \frac{d\Omega_R}{dR} := \lim_{R \to 0} \frac{\Omega_R}{R} = 2\phi_R \frac{d\phi_R}{dR} (l_{1,s} - l_{0,s}), \quad \frac{d\phi_R}{dR} := \frac{1}{||q||}. \]

According to L’Hôpital’s rule, the same result is attained by taking the derivative at \( R = 0 \). Hence, the choice of notation. We discuss the construction of Monte Carlo estimates in this setting below.

This derivation directly implies a strategy to sample linear lights proportional to solid angle, which is a limit case of prior work [Gam16]. Since the integrand in Equation (1) is constant, we simply sample \( \omega_\phi \) uniformly in \([l_{0,s}, l_{1,s}]\), set \( \omega_t := \sqrt{1 - ||\omega||^2} \) and return \( \omega_s \omega + \omega_t \) as sampled direction. Sec. 3.5 describes the implementation in more detail.
3.2. Sampling the Projected Solid Angle of a Line

To obtain the projected solid angle from the above formulation of the solid angle, we introduce a cosine term for the surface normal \( n \in \mathbb{S}^2 \). The normal has local coordinates \( n_0 := (s, n) \), \( n_1 := (t, n) \). Recall that the local coordinate \( \omega_p \in [0, \pi] \) corresponds to the normalized direction \( \omega_p s + \sqrt{1 - \omega_p^2} t \). Thus, the projected solid angle per radius for \( R \to 0 \) is

\[
\frac{d\Omega}{dR} := 2 \frac{d\phi_R}{dR} \int_{l_0}^{l_1} \left( n, \omega_p s + \sqrt{1 - \omega_p^2} t \right) d\omega_p
\]

\[
= 2 \frac{d\phi_R}{dR} \left( n_0 \omega_p + n_1 \sqrt{1 - \omega_p^2} d\omega_p \right)
\]

\[
= \frac{d\phi_R}{dR} \left( F_{n_0}(l_1, s) - F_{n_0}(l_0, s) \right),
\]

where we use the indefinite integral

\[
F_{n_0}(\omega_p) := n_0 \omega_p^2 + n_1 \left( \sqrt{1 - \omega_p^2} \omega_p + \arcsin \omega_p \right).
\]

The dot product in this integral must not be negative. Therefore, we clip the line connecting \( l_0 \) to \( l_1 \) against the tangent plane of the surface.

We intend to use inverse function sampling to sample this projected solid angle uniformly. Our sampling procedure consumes a single uniform random number \( \xi \in [0, 1] \). The sample coordinate \( \omega_p \) has to be chosen so that the value of the distribution function matches the random number, i.e.

\[
\frac{d\phi_R}{dR} \left( F_{n_0}(\omega_p) - F_{n_0}(l_0, s) \right) = \xi \frac{d\Omega}{dR}
\]

Assuming that we can evaluate the inverse distribution function \( F_{n_0}^{-1} \), the solution is

\[
\omega_p = F_{n_0}^{-1} \left( \xi \frac{d\Omega}{dR} \frac{d\phi_R}{dR} + F_{n_0}(l_0, s) \right).
\]

The whole sampling procedure, including optimizations described in Section 3.5, is summarized in Algorithm 1.

3.3. Inversion of the Distribution Function

Inversion of \( F_{n_0} \) is challenging. A closed-form solution appears to be impossible. Since scaling of \( (s, n) \) only scales the integral \( F_{n_0} \), we are dealing with a one-dimensional family of functions to invert. Fig. 3 shows examples. It is possible to use a two-dimensional lookup table but that requires a high resolution at boundaries and memory access at random locations thrashes caches. Instead, we take inspiration from recent work [Pet21] and design a specialized iterative procedure with rapid convergence.

![Figure 3: Plots of the inverse distribution function \( F_{n_0}^{-1} \) for five choices of \( n_0, n_1 \). At the left domain boundary, the derivative always approaches infinity. For \( n_0 \) near \(-1\) (orange, blue), the domain shrinks and the function becomes steep. \( F_{l_0}^{-1} \) (green) is simply a square root.](image)

For our iterative procedure, we make the substitution \( \alpha := \arcsin \omega_p \) and consider

\[
G_{n_0}(\alpha) := F_{n_0}(\sin \alpha) = n_0 \sin^2 \alpha + n_1 (\cos \alpha \sin \alpha + \alpha).
\]

Evaluation of this function does not involve costly inverse trigonometric functions and it is more well-behaved. In fact, quadratic Taylor expansions give good local fits of \( G_{n_0}(\alpha) \). Our iterative method exploits that. In each step, it constructs a quadratic Taylor polynomial around the current estimate of \( \alpha \) and solves Equation (4) with this approximation. The equation turns into a quadratic. Among the two solutions, we pick the one that is closer to the current estimate. If there are no roots, we take the extremum of the quadratic instead to safeguard against rare numerical issues.

This approach can be thought of as quadratic generalization of Newton’s method. It is known as Halley’s irrational formula or Laguerre’s method [ST95]. Laguerre’s method is popular for polynomial root finding but uncommon as general root finding method [PTVF07]. Sec. 3.4 demonstrates that it works well here.

For the initialization, we use solid angle sampling. As derived in Sec. 3.1, that means that we simply set \( \omega_p \) to

\[
l_0, \alpha + \xi (l_1, s - l_0, s).
\]

Thus, this initialization is extremely efficient. We proceed to show that it is also accurate enough. Algorithm 2 summarizes our inversion procedure.

3.4. Error Analysis

To avoid branch divergence on GPUs, we want to use a small, fixed iteration count. However, we also want our renderer to be unbiased. Thus, we have to be sure that our method converges to sufficient accuracy in all cases. Since the beginning and the end of the line influence the initialization, the set of all test cases is four-dimensional.

Once again, we take inspiration from recent work [Pet21]. We run the Nelder-Mead optimizer [NM65] in 80-bit float arithmetic to find a line that maximizes the error of our iterative procedure. Since
Nelder-Mead is just a local optimizer, we try 28 billion random initializations. Sometimes the iteration is unstable near the endpoints of the linear light. However, the initialization is nearly perfect there. Thus, we skip the iteration if $\xi < 10^{-5}$ or $\xi > 1 - 10^{-5}$.

As error metric, we use a backward error, namely the perturbation in the random number $\xi$ that suffices to explain the error in the output. This way, we find that the worst possible error after two iterations is $1.58 \cdot 10^{-5}$. On this basis, we consider our method unbiased.

In practice, rounding errors in single-precision arithmetic are far more influential than these theoretical errors (Fig. 4a). The main problem is a cancellation in Equation (2). The indefinite integral $F_{n,n_i}$ corresponds to the projected solid angle of a line starting at the closest point $q$. When the actual projected solid angle is much smaller, we lose precision. However, that only happens in dark regions along the infinite extension of the line. If we take that into account, errors are always low (Fig. 4b). Therefore, we choose not to invest computational resources to avoid this cancellation.

### 3.5. Optimizations

Conceptually, it is useful to work with the coordinate frame $s, t$ but in practice we skip computation of $t \in \mathbb{S}^2$. We only need it for two purposes: To compute $n_t$ and to construct directions $\omega_n s + \omega_t t$. In both cases, we exploit

$$ t = \frac{1}{\| q \|} (l_0 - (l_0, s) s). $$

Then

$$ \omega_n s + \omega_t t = \left( \frac{\omega_n}{\| q \|} (l_0, s) \right) s + \frac{\omega_t}{\| q \|} l_0, $$

$$ n_t = \frac{1}{\| q \|} ((n, l_0) - n_s (l_0, s)). $$

For efficiency reasons, lines are represented by their beginning $l_0 \in \mathbb{R}^3$, their direction $s \in \mathbb{S}^2$ and their length $L := \| l_1 - l_0 \|$. We precompute these attributes per linear light. Since $l_1 = l_0 + L s$, we have $(l_1, s) = (l_0, s) + L$.

Algorithms 1 and 2 implement our method with these optimizations. Our supplementary code additionally eliminates common subexpressions like $\langle l_0, s \rangle$, $\langle l_0, l_1 \rangle$ and $F_{n,n_i}(l_0,s)$. We always clamp coordinates $\omega_0$ to $[-1, 1]$ to avoid invalid results. Blinn’s quadratic solver [Bli06] makes Algorithm 2 more stable. Besides, it is useful to split Algorithm 1 into a part that executes once per line and another part that runs once per sample.

Minor changes turn Algorithm 1 into a heavily optimized implementation of solid angle sampling. We simply omit all lines that are trivial to avoid invalid results. Blinn’s quadratic solver [Bli06] makes Algorithm 2 more stable. Besides, it is useful to split Algorithm 1 into a part that executes once per line and another part that runs once per sample.

### 4. Sampling Linear Lights for Specular Shading

By itself, our projected solid angle sampling gives low variance for diffuse BRDFs but not for specular BRDFs, especially at low roughness (Fig. 1). The same is true for sampling of polygonal lights and we overcome this limitation in the same way [Pet21]. This section briefly describes the necessary steps. For a detailed discussion, we refer to prior work [Pet21].
Algorithm 2: invert_line_sampling_cdf

Input: \( n_0, n_1 \in \mathbb{R} \), an initialization \( \omega_0 \in [-1, 1] \), \( F \in \mathbb{R} \)

Output: \( \omega_k = F_{n_0, n_1}^{-1}(F) \in [-1, 1] \), \( \omega_t = \sqrt{1 - \omega_k^2} \)

\[ \alpha := \arcsin \omega_0 \]
\[ \omega_k := \sqrt{1 - \omega_k^2} \]

Repeat twice:

\[ G_0 := (n_0 \omega_0 + n_1 \omega_0) \omega_k + n_1 \alpha - F \quad // = G_{n_0, n_1}(\alpha) - F \]
\[ G' := 2(n_0 \omega_0 + n_1 \omega_0) \omega_k \]
\[ G'' := 2(n_0 \omega_0 - n_1 \omega_0) \omega_k - 2(n_0 \omega_0 + n_1 \omega_0) \omega_k \]

Solve \( G''/2 = \beta^2 + G' \beta + G_0 = 0 \)

Let \( \beta \) be the root of smaller magnitude (extremum if none exists)

\[ \alpha := \min \left( \max \left( \alpha + \beta, -\frac{\pi}{2}, \frac{\pi}{2} \right) \right) \]
\[ \omega_k := \sin \alpha \]
\[ \omega_t := \cos \alpha \]

Return \( \omega_k, \omega_t \)

The LTC density \( p_M \) is zero in parts of the upper hemisphere and non-zero in parts of the lower hemisphere.

The LTC density \( p_M \) is zero in parts of the upper hemisphere and non-zero in parts of the lower hemisphere.

Let \( T \) be a transformation that changes the opening angle of the cylinder. When using LTCs for linear lights, there is a potential pitfall: Linear transformations change the opening angle of the cylinder. To account for this effect, we use the correction factor \cite{HH17a}

\[ \frac{dR}{dR_c} := \frac{\| M^T (s \times l_0) \|}{\| s \times l_0 \|}. \]

We multiply it onto densities and divide it out of projected solid angles and LTC shading estimates.

4.2. Combining Diffuse and Specular Samples

The LTC density \( p_M \) is zero in parts of the upper hemisphere (Fig. 5). Thus, we have to combine the corresponding samples with samples from projected solid angle sampling to get an unbiased estimate. Standard MIS heuristics introduce considerable variance outside of penumbrae. For example, the balance heuristic effectively samples the sum of both densities but this density differs from the BRDF times cosine. Therefore, we use clamped optimal MIS \cite{Pet21}, which is designed specifically for this situation.

To use it, we have to compute estimates of unshadowed diffuse and specular shading \( c_0, c_1 \) as in the original work on LTCs \cite{HDHN16, HH17a}. Both of these are computed per color channel using the readily available projected solid angle of the linear light.

All entries of \( c_0 \) must be non-zero, so we clamp diffuse albedos to a minimum of 0.01. Then clamped optimal MIS weights are \cite{Pet21}

\[ w_j(\omega_j) := \frac{c_j p_j(\omega_j) dR}{\sum_{k \in [0, 1]} p_k(\omega_k) dR}, \]

\[ \sum_{k \in [0, 1]} p_k(\omega_k) dR \]

where \( j \in \{0, 1\} \) is the index of the technique that generated the sample \( \omega_j \in \mathbb{S}^2 \) and \( p_k(\omega) dR \) is the density times radius for technique \( k \). The parameter \( \nu \in [0, 1] \) blends between the standard balance heuristic and a weighted balance heuristic, which is optimal under idealizing assumptions such as no occlusion \cite{Pet21}. Setting \( \nu = 0.5 \) works well in practice.

5. Results

In the following, we evaluate the quality of our importance sampling for diffuse (Sec. 5.1) and specular shading (Sec. 5.2) in comparison to prior work. We also make recommendations on stratification (Sec. 5.3) and measure timings (Sec. 5.4).

Our Vulkan renderer uses the extension VK_KHR_ray_query to cast shadow rays. It is a deferred renderer with a 32-bit visibility buffer \cite{BH13}. Unless stated otherwise, our experiments use the isotropic Frostbite BRDF \cite{LdR14}. For LTCs, we use a 64 \( \times \) 64 \( \times \) 51 table of transforms \( M \) parameterized by roughness, outgoing inclination and Fresnel reflectance at 0°. Support for arbitrary anisotropic BRDFs would require a 5D table, which is hardly viable. We inherit this limitation from LTCs \cite{HDHN16}. Linear lights are displayed with finite extent to convey geometric relations better. Alongside our results, we report root-mean-square errors (RMSEs) computed from HDR frames.
5.1. Diffuse Shading

Figure 6 compares different approaches for diffuse shading using a Lambertian diffuse BRDF. Area sampling (Fig. 6a) places samples uniformly along the length of the linear light. The square falloff term and the two cosine terms introduce strong variance, especially on the ceiling and near the light. These regions look darker because sRGB values get clamped at one. Solid angle sampling (Fig. 6c) is far better but the remaining cosine term still causes variance, especially on the white wall, where it ranges down to zero. On the box (orange inset) the linear light is partially below the horizon. Clipping (Fig. 6d) eliminates samples without contribution.

We apply warping of random numbers [HPM+20] on top of clipped solid angle sampling to incorporate the cosine term into the density. With a linear density, this approach is effective on the back wall but barely improves results on the ceiling or the red wall (Fig. 6e). Results deteriorate in the overexposed parts of the ceiling, hence the bad RMSE. A quadratic density helps everywhere, at an increased overhead (Fig. 6f).

As expected, our projected solid angle sampling achieves zero variance outside of penumbrae (Fig. 6h). Noise in penumbrae is not reduced significantly but stratification through blue noise works well (red inset). The method of Li et al. [LADL18] gives identical results at a higher cost.

5.2. Specular Shading

Fig. 1 demonstrates the benefits of our specular importance sampling. The puddle in the foreground (red inset) has low roughness such that solid angle sampling and projected solid angle sampling rarely sample the peak of the specular BRDF. Thus, shading is far from convergence at one sample per pixel. Using an additional specular sample distributed proportional to an LTC through clamped optimal MIS with \( v = 0.5 \) improves the result drastically. Remaining noise is mostly due to the penumbra of the bollard. Note that the shadow of the bollard looks more like a glossy reflection due to the narrow peak of the BRDF.

5.3. Stratification

We find two established methods to be particularly effective for linear lights. Ramamoorthi et al. [RAMN12] recommend uniform jittered sampling for linear lights. That means that we only consume a single random number \( \xi \) on \([0, 1)\) per technique per light. If we take \( N \in \mathbb{N} \) samples, the random number fed to sampling algorithms for sample \( k \in \{0, \ldots, N-1\} \) is \( \frac{k}{N} \). Indeed, this approach gives an appreciable reduction of variance in penumbrae (Fig. 7). Additionally, we use precomputed \( 64 \times 64 \) blue noise textures [Uli93]. The blue noise patterns are preserved relatively well in penumbrae.

5.4. Run Time

To measure timings, we use the same setup as previous work [Pet21] such that numbers are comparable. Our test system consists of an NVIDIA GeForce RTX 2080 Ti, an Intel Core i5-9600K and 16 GB RAM. When ray tracing is enabled, our clipped solid angle sampling and our projected solid angle sampling perform identically. The computation is hidden by the latency, even...
with the geometrically simple Cornell box (Fig. 6). Adding one sample per pixel at 1920 × 1080 resolution for the bistro exterior (Fig. 1) takes 0.45 ms.

Therefore, we focus on computational cost and disable ray tracing. We point the camera at a plane and either take 128 samples from 128 different lights or all from the same light. Our renderer has an overhead per sample, e.g., to evaluate the BRDF. To measure this overhead separately, we define a cheap baseline sampling technique, namely area sampling without proper density computation.

Table 1 lists the results. Our optimized solid angle sampling is extremely fast, especially regarding the cost per sample. Note however, that the marginal cost of the baseline sampling technique comes on top of that. Even area sampling has a higher cost per sample due to the more complex density. Clipping doubles the cost per light. Warping [HPM20] benefits from our fast implementation of solid angle sampling. The quadratic variant with closed-form cubic solver performs similarly to our projected solid angle sampling. The linear variant is faster but both of these have inferior quality.

Compared to the method with Newton bisection [LADL18], our projected solid angle sampling costs 2.5 times less per sample and 2 times less per light. However, Newton bisection has a variable iteration count and potentially divergent execution. Therefore, we repeat this experiment on the geometrically more complex scene in Fig. 1 and find that our cost per sample is 4.5 times lower. This gap could grow further in a full path tracer.

Through rejection sampling, rectangle sampling techniques can sample cylinders [Gam16]. However, our specialized methods for the limit case are considerably faster for solid angle sampling [UnFK13] and projected solid angle sampling [Pet21], respectively.

6. Conclusions

Our work takes importance sampling of linear lights to its natural conclusion. Except for visibility, all terms of the reflection equation are accounted for and the method is stable and inexpensive. It is a valuable addition to any path tracer and also offers efficiency improvements for differentiable rendering [LADL18]. Methodically, our work reinforces the value of tailor-made iterative algorithms for sampling problems in computer graphics. Since we guarantee accurate results with two inexpensive iterations, there is no practical reason to prefer closed-form solutions.

Now that fast projected solid angle sampling and LTC importance sampling are available for polygonal [Pet21] and linear lights, the most important remaining light type are ellipsoids. Spheres have been addressed [PD19] but that is not sufficient for LTC importance sampling. We hope that similar iterative methods will be applicable. Our optimizations may also apply to rectangle solid angle sampling [UnFK13]. Besides, our work further motivates the generalization of LTCs to arbitrary anisotropic BRDFs.

Acknowledgments

We thank Alisa Jung, Johannes Schudeiske, Tobias Zirr, Carsten Dachsbacher and our reviewers for their constructive input. Figs. 1 and 7 use ORCA models courtesy of Amazon Lumberyard. Our renderer ships with additional scenes. The attic is created by Shuprobho Das, Fran Calvente, Asteril, YopLand, M. Ziemys, Eva11dragon, Rob Tuytel and me. The arcade is made by Nicholas Hull for NVIDIA’s Falcor. The living room is made by Jay Artist.
References


© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.