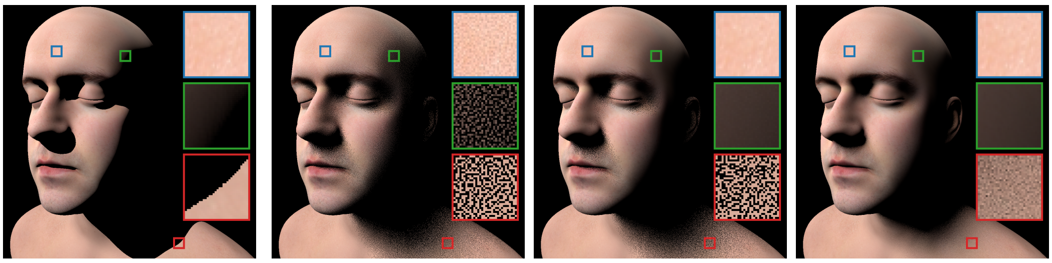
Sampling Projected Spherical Caps in Real Time

CHRISTOPH PETERS, Karlsruhe Institute of Technology, Germany
CARSTEN DACHSBACHER, Karlsruhe Institute of Technology, Germany



(a) Point light (1 sample per pixel, 0.27 ms)
(b) Solid angle sampling (1 sample per pixel, 0.29 ms)
(c) Our projected solid angle sampling (1 sample per pixel, 0.30 ms)
(d) Our projected solid angle sampling (32 samples per pixel, 4.4 ms)

Fig. 1. Spherical light sources (b,c,d) give a more pleasant and smooth shading than point lights (a). However, sampling a spherical light source proportionally to solid angle introduces noise, especially for lighting at grazing angles. Our projected solid angle sampling produces nearly noise-free results outside of penumbrae. An adequate sampling of the visibility in penumbrae still requires more samples. Timings are full frame times for a resolution of 1024^2 on an NVIDIA RTX 2080 Ti and we use the Disney diffuse BRDF with roughness 0.5.

Stochastic shading with area lights requires methods to sample the light sources. For diffuse materials, the best strategy is to sample proportionally to projected solid angle. Recent work in offline rendering has addressed this problem for spherical light sources, but the solution is unsuitable for a GPU implementation. We present a far more efficient solution. It offers results without noteworthy noise for diffuse surfaces lit by an unoccluded spherical light source while being only two to three times more costly than simple sampling of the solid angle. The core insight of the technique is that a projected spherical cap can be decomposed into, or at least approximated by, cut disks. We present an efficient method to sample cut disks and show how to use it to sample projected spherical caps. In some cases, our method does not sample exactly proportionally to projected solid angle but the deviation is provably bounded.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: sampling, projected spherical caps, spherical light sources, area lights, diffuse shading, real-time shadows, ray traced soft shadows

Authors' addresses: Christoph Peters, christoph.peters@kit.edu, Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131, Karlsruhe, Germany; Carsten Dachsbacher, dachsbacher@kit.edu, Karlsruhe Institute of Technology, Karlsruhe, Germany.

ACM Reference Format:
Christoph Peters and Carsten Dachsbacher. 2019. Sampling Projected Spherical Caps in Real Time. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 1, Article 1 (May 2019), 16 pages. https://doi.org/10.1145/3320282

1 INTRODUCTION

In offline rendering, spherical light sources are the most common light type besides emissive geometry. They approximate many real-world light sources well (e.g. the sun or light bulbs) and provide pleasantly smooth shading as well as plausible soft shadows. Lately, these positive traits have lead to an increased interest in spherical lights for real-time rendering [Dupuy et al., 2017].

Meanwhile, advances in graphics hardware have made real-time ray tracing viable. Distributed ray tracing for soft shadows is among the first applications of this new hardware. Hence, shading is no longer computed through a closed-form solution to an integral but approximated by means of Monte Carlo integration. This approach requires a stochastic sampling of light sources.

It is well-known that the sampling strategy has a major influence on the variance in the resulting image. A good importance sampling strategy ensures that parts of the light source with a larger contribution to the reflected radiance are sampled more frequently. For diffuse surfaces, this means that parts of the spherical light source should be sampled proportionally to their projected solid angle. Since a diffuse BRDF is nearly constant, results of this strategy show almost no noise outside of penumbra regions.

Ureña and Georgiev [2018] have recently presented the first practical offline-rendering technique to sample in this manner. They demonstrate an impressive quality improvement at a cost that is reasonable in relation to the cost of ray tracing. Furthermore, the technique is compatible with stratification as it only takes two random numbers as input. However, it is not well-suited for graphics hardware. It uses inverse function sampling, but performs this inversion by means of Newton's method or a fallback to bisection. Performance of such iterative methods on graphics hardware severely suffers from branch divergence. Beyond that, numerical instabilities necessitate the use of double-precision arithmetic.

In this paper, we present a far more efficient solution: It is not iterative, does not require lookup tables and works robustly in single-precision arithmetic. It is based on a novel method to sample a cut disk uniformly. Like previous work, it is compatible with stratification and depends continuously on the input. In most cases, the resulting density is exactly proportional to the projected solid angle. Only when the center of the light source is below the horizon, we allow a provably bounded deviation in the density. Since we can compute the density exactly, the Monte Carlo estimate remains unbiased but its variance is slightly higher than with more computationally expensive solutions [Ureña and Georgiev, 2018].

We validate our technique by rendering ray traced soft shadows for spherical lights on diffuse surfaces in real time. Our results show almost no noise outside of penumbra regions (Fig. 1). The quality improvement compared to the less intricate solid angle sampling is greatest for lighting at grazing angles and with large light sources. Our technique is two to three times more expensive than basic solid angle sampling, which is a minor part of the shading work anyway. We also compare our approach to the work of Ureña and Georgiev [2018] and find that it is substantially faster without significant differences in quality. Full source code for our renderers is made available.

2 RELATED WORK

The reflection equation is naturally written as an integral over solid angle. To avoid introducing additional geometry terms with additional variance, Monte Carlo integrators should prefer sampling proportionally to the solid angle rather than the area of light sources. Therefore, sampling of various primitives proportionally to their solid angle is a well-studied problem. Uniform samples from the solid angle of a sphere, commonly referred to as spherical cap, are obtained by sampling the cosine between its center vector and the sampled direction uniformly [Wang, 1992]. The solid angle of triangles can be sampled approximately [Wang, 1992] or exactly [Arvo, 1995] with efficient closed forms. Polygons may be represented by several triangles but there is also a specialized solution for rectangles [Ureña et al., 2013]. Tight approximations of disks and cylinders by rectangles lead to a sampling strategy for these primitives [Gamito, 2016]. Alternatively, the solid angle of disks and other primitives such as ellipsoids that map to spherical ellipses can be sampled directly [Guillén et al., 2017, Heitz, 2017].

Sampling the projected solid angle yields even better results (see Section 3.1) but work on this topic is less extensive. For triangles, there is an approach based on recursive subdivision [Ureña, 2000]. A closed form to compute (but not sample) the projected solid angle of spheres has been known for a long time [Snyder, 1996]. More recent work [Ureña and Georgiev, 2018] addresses the sampling of projected spherical caps, just like our work. Since this work is closely related, we discuss commonalities and differences throughout the paper.

When used for the sampling of light sources, these techniques are part of a larger system. Stratified sampling guarantees that certain subsets are sampled equally to accelerate convergence [Christensen et al., 2018]. Many of the sampling techniques above are compatible with stratification as they use exactly two (quasi-)random numbers to produce one sample [Arvo, 1995, Guillén et al., 2017, Ureña et al., 2013, Ureña and Georgiev, 2018, Wang, 1992]. Sampling among the hundreds of thousands of lights in offline rendering necessitates the use of light hierarchies [Conty Estevez and Kulla, 2018]. For spherical light sources and specular BRDFs, the product of an approximate specular BRDF and the light source can be sampled exactly [Dupuy et al., 2017]. Related work has shown how to approximate specular BRDFs to compute lighting from polygonal and spherical light sources in closed form [Heitz, 2017, Heitz et al., 2016].

Stochastic real-time rendering needs to work with low sample counts and thus denoising is essential. To ensure that denoising is only needed in penumbra regions, Heitz et al. [2018] use a ratio estimator that maintains additional screen space buffers and evaluates exact shading without shadows. Spatial filtering with adaptive filter weights, combined with temporal accumulation, leads to a filter that is applicable in real time and produces noise-free results from input images path traced at one sample per pixel [Schied et al., 2017]. This filter introduces more blur for noisy input such that good importance sampling remains valuable.

3 SAMPLING PROJECTED SPHERICAL CAPS

In the following, we develop our method to sample projected spherical caps. We begin by explaining why such a strategy reduces variance. Then we analyze the geometry of spherical caps and introduce solutions for three different geometric configurations. Their common building block is a novel way to sample cut disks. Finally, we combine all solutions into a single sampling procedure.

3.1 A Radiometric Motivation

Our primary motivation is to approximate the lighting from a spherical light source. Consider a sphere with center $\boldsymbol{q} \in \mathbb{R}^3$ and radius $r > 0$ emitting light isotropically with a radiance L_e at all surface points. This light source illuminates a surface element at position $\boldsymbol{p} \in \mathbb{R}^3$ with unit normal $\boldsymbol{n} \in \mathbb{S}^2$ and BRDF f.

According to the reflection equation, we have to integrate over the solid angle of the light source, which forms a spherical cap on the unit sphere $\mathbb{S}^2$:

$$\mathbb{D} := \left\{ \frac{\boldsymbol{s} - \boldsymbol{p}}{\|\boldsymbol{s} - \boldsymbol{p}\|_2} \mid \boldsymbol{s} \in \boldsymbol{q} + r\mathbb{S}^2 \right\} \subseteq \mathbb{S}^2$$

The reflection equation states that the radiance reflected towards direction $\boldsymbol{\omega}_o \in \mathbb{S}^2$ is

$$L_o(\boldsymbol{p}, \boldsymbol{\omega}_o) = L_e \int_{\mathbb{D}} f(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i) V(\boldsymbol{p}, \boldsymbol{s}(\boldsymbol{\omega}_i)) \max\{0, \boldsymbol{n}^\mathsf{T} \boldsymbol{\omega}_i\} \, \mathrm{d}\boldsymbol{\omega}_i, \tag{1}$$

where $\boldsymbol{s}(\boldsymbol{\omega}_i) \in \mathbb{R}^3$ denotes the point where the ray $\boldsymbol{p} + t\boldsymbol{\omega}_i$ first intersects the sphere and the visibility V is one if the given line segment is unoccluded and zero otherwise.

Equation (1) suggests that a Monte Carlo integrator should sample the spherical cap $\mathbb{D}$ uniformly but this approach is not ideal. The visibility $V(\boldsymbol{p}, \boldsymbol{s}(\boldsymbol{\omega}_i))$ is constant outside of penumbrae and when the BRDF is diffuse, f is nearly constant. Then the Lambert term $\max\{0, \boldsymbol{n}^\mathsf{T} \boldsymbol{\omega}_i\}$ is the primary source of variation in the integrand. Light sources that are partially below the horizon lead to samples with no contribution. Even for light sources entirely above the horizon, some samples may have a much larger contribution than others. Lighting at grazing angles and large light sources provoke this problem. An alternative strategy would sample the area of the light source uniformly but that would introduce an additional directional cosine and a square falloff term, thus making matters worse.

To eliminate all geometric terms at once, we rewrite the integral as an integral over the projected solid angle of the light source. The projected spherical cap is the projection of the parts of the spherical cap above the horizon to the plane orthogonal to the normal $\boldsymbol{n}$:

$$\mathbb{P} := \{\boldsymbol{\omega} - (\boldsymbol{n}^\mathsf{T}\boldsymbol{\omega})\boldsymbol{n} \mid \boldsymbol{\omega} \in \mathbb{D} \text{ with } \boldsymbol{n}^\mathsf{T}\boldsymbol{\omega} \geq 0\} \subseteq \mathbb{R}^3$$

By the Nusselt analog, this allows us to rewrite the reflection Equation (1) as

$$L_o(\boldsymbol{p}, \boldsymbol{\omega}_o) = L_e \int_{\mathbb{P}} f(\boldsymbol{\omega}_o, \boldsymbol{\omega}(\boldsymbol{\omega}_i^\perp)) V(\boldsymbol{p}, \boldsymbol{s}(\boldsymbol{\omega}(\boldsymbol{\omega}_i^\perp))) \, \mathrm{d}\boldsymbol{\omega}_i^\perp$$

$$\text{where} \quad \boldsymbol{\omega}(\boldsymbol{\omega}_i^\perp) := \boldsymbol{\omega}_i^\perp + \sqrt{1 - \|\boldsymbol{\omega}_i^\perp\|_2^2}\,\boldsymbol{n}$$

undoes the orthogonal projection. A Monte Carlo integrator that approximates this integral by sampling the projected spherical cap $\mathbb{P}$ uniformly (i.e. proportionally to projected solid angle) has a very low variance for diffuse BRDFs outside of penumbrae. Therefore, we wish to perform this sampling efficiently.

3.2 The Geometry of Projected Spherical Caps

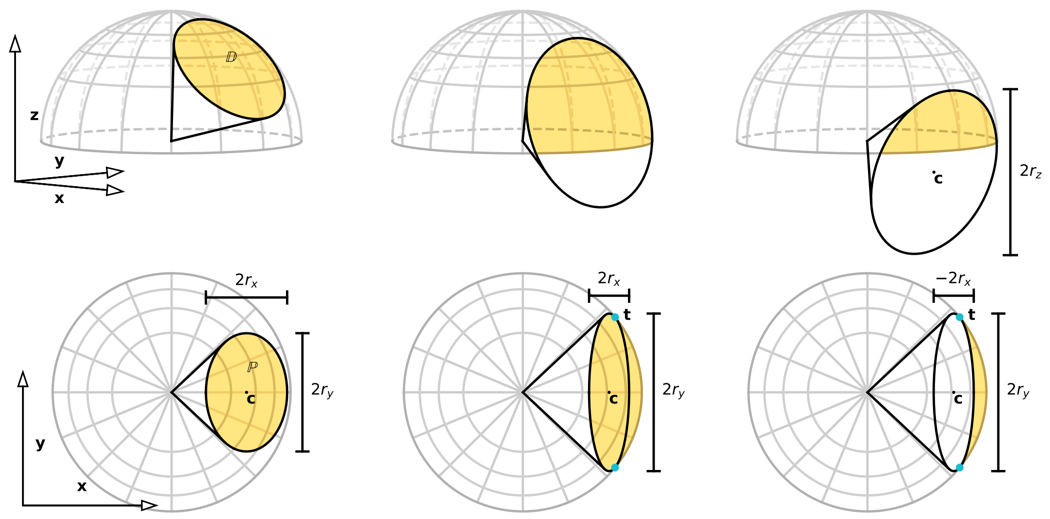
We begin by analyzing the shape of the projected spherical cap $\mathbb{P}$ and introducing a case distinction. An equivalent analysis is the starting point of Ureña and Georgiev [2018] although our formulation avoids the introduction of angles to save costly evaluations of trigonometric functions.

The boundary of the spherical cap $\partial\mathbb{D}$ is a circle on the unit sphere. Projecting this circle to the plane orthogonal to $\boldsymbol{n}$ yields an ellipse (Fig. 2a). To characterize this ellipse, we fix an orthonormal coordinate frame: The z-axis is aligned with the surface normal $\boldsymbol{n}$. The x-axis is aligned with the vector towards the light center $\boldsymbol{q} - \boldsymbol{p}$ and orthogonal to $\boldsymbol{n}$. Then the ellipse is entirely in the xy-plane and just like the circle it is symmetric about the xz-plane.

This ellipse characterizes the projected spherical cap $\mathbb{P}$ but a case distinction is needed to make their relation precise.

Case 1. If the spherical cap is entirely above the horizon, the mapping from the xy-plane to the spherical cap is one to one. Hence, $\mathbb{P}$ agrees with the ellipse (Fig. 2a).

Case 2. If the spherical cap intersects the horizon but has its center above it, the same can be said of the circle $\partial\mathbb{D}$. The ellipse is a subset of the projected spherical cap because projecting points inside the circle $\partial\mathbb{D}$ along z to the upper hemisphere yields points in the spherical cap. Beyond that, the spherical cap goes all the way down to the horizon. The horizon is the unit circle in the



(a) Case 1: Entirely above (b) Case 2: Mostly above (c) Case 3: Mostly below

Fig. 2. The shape of the projected spherical cap $\mathbb{P}$ (shaded yellow in the bottom row) depends on whether it intersects the horizon and whether its center is above the horizon. Spherical caps entirely below the horizon are not shown.

xy-plane and thus the ellipse is tangential to the unit circle at the two points where $\partial\mathbb{D}$ intersects the horizon. All points enclosed between the horizon and the line connecting these tangent points are also part of the projected spherical cap $\mathbb{P}$ (Fig. 2b).

Case 3. If the spherical cap intersects the horizon but has its center below it, the ellipse is still tangential to the unit circle at two points. This time, projecting points inside the circle $\partial\mathbb{D}$ along z onto the upper hemisphere yields points outside the spherical cap. Thus, only points enclosed between the horizon and the ellipse are part of the projected spherical cap $\mathbb{P}$ (Fig. 2c). The ellipse itself is not contained.

Case 4. The spherical cap is entirely below the horizon. Then the projected solid angle is empty. This case should be detected and reported back without producing a sample.

Algorithm 1 constructs the local coordinate frame and computes the center $\boldsymbol{c}$ and extent $\boldsymbol{r}$ of the circle $\partial\mathbb{D}$ along each axis. These quantities simultaneously characterize the ellipse in the xy-plane. It also computes one point $\boldsymbol{t}$ where the ellipse is tangential to the unit circle. We follow the convention that the extent r_x is negative in Case 3 or 4 and that $t_x \geq 1$ in Case 1 or 4. Thus, all cases can be identified without additional flags. The elementary geometric constructions that lead to Algorithm 1 and subsequent algorithms are detailed in the supplementary document.

3.3 Caps Mostly Above the Horizon (Case 2)

To derive our sampling procedure, we first consider Case 2 because it inspires the design decisions for all three cases. Ureña and Georgiev [2018] solve this case with inverse function sampling using iterative methods for the inversion.

We obtain a more efficient scheme by cutting the projected spherical cap $\mathbb{P}$ vertically along the line connecting the points where the ellipse is tangential to the unit circle (Fig. 3). The left part is a

Algorithm 1 `ComputeGeometry`
Input: Surface normal $\boldsymbol{n} \in \mathbb{S}^2$, vector to the center of a sphere $\boldsymbol{d} := \boldsymbol{q} - \boldsymbol{p} \in \mathbb{R}^3$ and radius $r > 0$.
Output: The coordinate frame $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{S}^2$, half the extent of the circle $\partial\mathbb{D}$ along each axis $\boldsymbol{r} \in \mathbb{R}^3$, the center of this circle $\boldsymbol{c} \in \mathbb{R}^3$ and the location of one of the two tangent points $\boldsymbol{t} \in \mathbb{R}^3$.

```
$\boldsymbol{\omega}_d := \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_2}$
$\boldsymbol{z} := \boldsymbol{n}$
$\boldsymbol{x} := \boldsymbol{\omega}_d - (\boldsymbol{z}^\mathsf{T}\boldsymbol{\omega}_d)\boldsymbol{z},\ u := \frac{1}{\|\boldsymbol{x}\|_2},\ \boldsymbol{x} := u\boldsymbol{x}$
$\boldsymbol{y} := \boldsymbol{z} \times \boldsymbol{x}$
$r_y := \frac{r}{\|\boldsymbol{d}\|_2},\ r_x := (\boldsymbol{z}^\mathsf{T}\boldsymbol{\omega}_d)r_y,\ r_z := (\boldsymbol{x}^\mathsf{T}\boldsymbol{\omega}_d)r_y$
$v := \sqrt{1 - r_y^2},\ c_x := (\boldsymbol{x}^\mathsf{T}\boldsymbol{\omega}_d)v,\ c_z := (\boldsymbol{z}^\mathsf{T}\boldsymbol{\omega}_d)v$
$t_x := uv,\ t_y := \sqrt{1 - t_x^2}$
Return $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, (r_x, r_y, r_z)^\mathsf{T}, (c_x, 0, c_z)^\mathsf{T}, (t_x, t_y, 0)^\mathsf{T}$
```

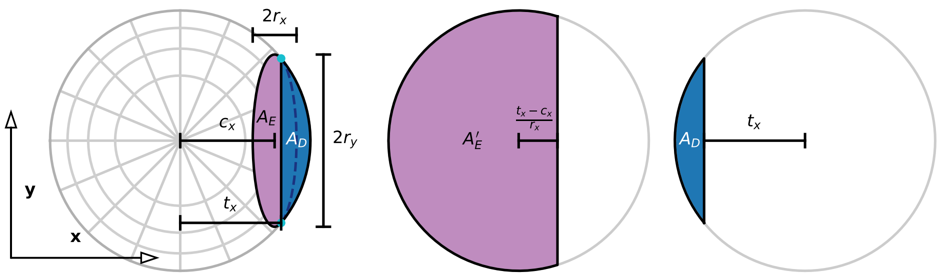


Fig. 3. If the spherical cap intersects the horizon but has its center above it, we decompose the projected spherical cap $\mathbb{P}$ into a cut ellipse and a cut unit disk. Through translation and scaling, they become two cut unit disks.

cut version of the ellipse, whereas the right part is a cut unit disk. Scaling turns the cut ellipse into another cut unit disk. Thus, we only need a method to sample a cut unit disk to sample $\mathbb{P}$.

We sample the x-coordinate first using inverse function sampling. The area of a unit disk in the strip between zero and $X \in [0, 1]$ is given by

$$A(X) := \int_0^X 2\sqrt{1 - x^2}\,\mathrm{d}x = X\sqrt{1 - X^2} + \arcsin X.$$

To perform inverse function sampling, we need to evaluate this function and its inverse. The former is trivial. For the inverse, we introduce an efficient approximation with high accuracy along with the remainder of the sampling procedure in Section 3.6.

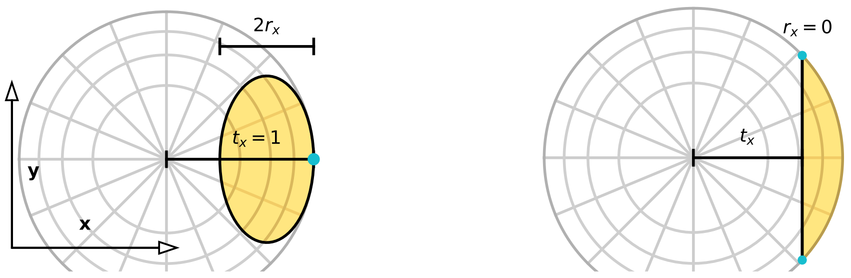
Now to sample the projected solid angle $\mathbb{P}$, we first compute the areas of the two cuts $A_E, A_D > 0$. A threshold on the first random number ξ_0 decides which of the two cuts needs to be sampled. Then the appropriate cut disk is sampled and the result is transformed back to the local coordinate frame. For Monte Carlo integration, we also need to compute the density of the sampled planar probability distribution. Since we are sampling uniformly, it is $(A_E + A_D)^{-1}$. All of this is done by Algorithm 2, which is constructed carefully to ensure a continuous dependence of the outputs on all inputs.

Algorithm 2 `MostlyAbove` (Case 2)
Input: $\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{t}$ from `ComputeGeometry` and random numbers $\xi_0, \xi_1 \in [0,1]$.
Output: Local sampled direction $\boldsymbol{\omega}_i'$ and density $p(\xi_0, \xi_1)$.

```
$A_D := \frac{\pi}{2} - A(t_x),\ A_E' := \frac{\pi}{2} + A\left(\frac{t_x - c_x}{r_x}\right),\ A_E := r_x r_y A_E'$
$p(\xi_0, \xi_1) := \frac{1}{A_E + A_D}$
If $\xi_0 < \frac{A_E}{A_E + A_D}$:        // Sample the cut ellipse
    $d_x, d_y :=$ SampleCutUnitDisk($\xi_0 \frac{A_E + A_D}{A_E} A_E', \xi_1$)
    $\omega_{i,x} := r_x d_x + c_x,\ \omega_{i,y} := r_y d_y$
Else:                                   // Sample the mirrored, cut unit disk
    $d_x, d_y :=$ SampleCutUnitDisk($(A_E + A_D)(1 - \xi_0), \xi_1$)
    $\omega_{i,x} := -d_x,\ \omega_{i,y} := d_y$
Return $\left(\omega_{i,x}, \omega_{i,y}, \sqrt{1 - \omega_{i,x}^2 - \omega_{i,y}^2}\right)^\mathsf{T}, p(\xi_0, \xi_1)$
```

Algorithm 3 `EntirelyAbove` (Case 1)
Input: $\boldsymbol{r}, \boldsymbol{c}$ from `ComputeGeometry` and random numbers $\xi_0, \xi_1 \in [0,1]$.
Output: Local sampled direction $\boldsymbol{\omega}_i'$ and density $p(\xi_0, \xi_1)$.

```
$A_E := r_x r_y \pi,\ p(\xi_0, \xi_1) := \frac{1}{A_E}$
$d_x, d_y :=$ SampleCutUnitDisk($\pi \xi_0, \xi_1$)
$\omega_{i,x} := r_x d_x + c_x,\ \omega_{i,y} := r_y d_y$
Return $\left(\omega_{i,x}, \omega_{i,y}, \sqrt{1 - \omega_{i,x}^2 - \omega_{i,y}^2}\right)^\mathsf{T}, p(\xi_0, \xi_1)$
```

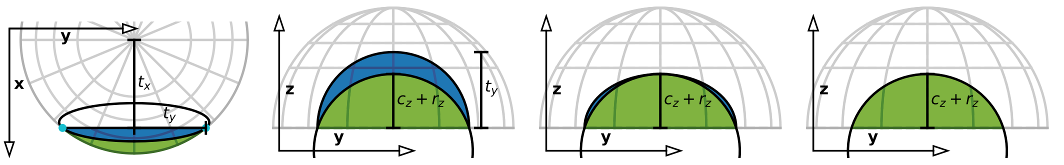


(a) Tangent to the horizon (between Cases 1 and 2) (b) Center on the horizon (between Cases 2 and 3)

Fig. 4. How our method achieves continuity across cases. If the spherical cap is tangent to the horizon, sampling the entire ellipse gives the same result as sampling a full ellipse and an empty cut disk. If the center is on the horizon, sampling an empty ellipse and a cut disk gives the same result as sampling the cut disk and applying an identity warp.

3.4 Caps Entirely Above the Horizon (Case 1)

If the spherical cap is entirely above the horizon, we only need to sample an ellipse uniformly. The standard approach would be to sample the unit disk in polar coordinates and to scale the result to an ellipse as is done in the radial approach of Ureña and Georgiev [2018]. In our case, this solution would lead to a discontinuity when the spherical cap begins to intersect the horizon.



(a) Top view of the cut disk (b) Projected into the yz-plane (c) Scaled uniformly along z (d) Scaled non-uniformly along y

Fig. 5. The steps used to warp the cut disk onto the projected spherical cap $\mathbb{P}$. The warped, cut disk is shown in blue, areas where it overlaps the projected spherical cap $\mathbb{P}$ are green. The cut disk is projected onto the hemisphere and into the yz-plane (b), the resulting semicircle is scaled uniformly along the z-axis to make its height match the spherical cap (c) and then scaled non-uniformly along the y-axis to bring it into full agreement (d).

To avoid this discontinuity and to have fewer instructions that execute in one branch only, we reuse our method to sample cut disks. The disk is not actually cut as it corresponds to the entire ellipse (Fig. 4a). Algorithm 3 implements this method.

Sampling in terms of polar coordinates is slightly faster (see Section 4.4) but the difference is small enough that we prefer to avoid the discontinuities as they may ruin stratification across pixels and violate assumptions of some algorithms.

3.5 Caps Mostly Below the Horizon (Case 3)

Uniform sampling of the lune formed between the ellipse and the unit disk is a difficult problem. Unlike the cut disk, the lune depends on two independent parameters c_x and r_x. Ureña and Georgiev [2018] address this case by iterative inversion. We have not found an exact closed-form solution either. Nonetheless, we avoid a compromise on efficiency since we target a real-time setting. Instead we compromise quality to a small extent by allowing a bounded deviation from perfectly uniform sampling.

We once more sample a cut disk and then we warp it onto the lune. This warp is carefully designed to avoid unbalanced changes in area. Since the corresponding density is computed exactly, it does not introduce bias but does increase the variance slightly.

We begin by sampling the cut disk given by the tangent points uniformly (Fig. 5a). To understand the functioning of the warp, it is useful to project the sampled cut disk and the lune back onto the hemisphere and then into the yz-plane. The cut disk maps to a semicircle while the lune is a cut ellipse (Fig. 5b). We scale the cut disk uniformly along the z-axis such that it has the same height as the lune (Fig. 5c). Then we scale y-coordinates dependent on z to bring the cut disk into full agreement with the lune (Fig. 5d). Finally, we compute x such that the sampled direction has unit length again.

Note that the cut disk agrees with the lune when the center of the spherical cap is exactly on the horizon (Fig. 4b). In this case, the warp is the identity and the behavior across cases is continuous once more. Algorithm 4 implements our strategy.

While the construction above may seem peculiar, it is the only one among many tested alternatives that gives a bounded variation for the probability density function. In the supplementary document, we prove that for a fixed spherical cap

$$\frac{\max_{\xi_0,\xi_1\in[0,1]} p(\xi_0,\xi_1)}{\min_{\xi_0,\xi_1\in[0,1]} p(\xi_0,\xi_1)} \leq \frac{c_x - r_x}{t_x}\sqrt{\frac{t_x-(c_x+r_x)}{-r_x}}.$$

Algorithm 4 `MostlyBelow` (Case 3)
Input: $\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{t}$ from `ComputeGeometry` and random numbers $\xi_0, \xi_1 \in [0, 1]$.
Output: Local sampled direction $\boldsymbol{\omega}'_i$ and density $p(\xi_0, \xi_1)$.

```
$A_D := \frac{\pi}{2} - A(t_x)$
$d_x, d_y :=$ SampleCutUnitDisk($A_D(1 - \xi_0), \xi_1$)
$d_z := \sqrt{1 - d_x^2 - d_y^2}$
$\omega_{i,z} := \frac{c_z + r_z}{t_y} d_z$
$s_y := r_y \sqrt{\frac{r_z^2 - (\omega_{i,z} - c_z)^2}{r_z^2 (t_y^2 - d_z^2)}}, \ \omega_{i,y} := s_y d_y$
$\omega_{i,x} := \sqrt{1 - \omega_{i,y}^2 - \omega_{i,z}^2}$
$p(\xi_0, \xi_1) := -\frac{t_y^2}{(c_z + r_z)^2 A_D} \frac{\omega_{i,x}}{d_x s_y}$
Return $(\omega_{i,x}, \omega_{i,y}, \omega_{i,z})^\mathsf{T}, p(\xi_0, \xi_1)$
```

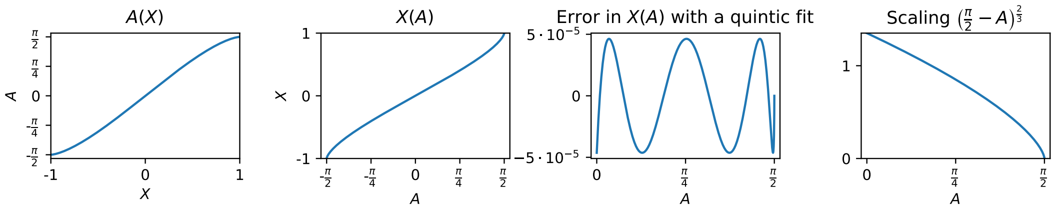


Fig. 6. The function $A(X)$, its inverse, the error of our approximation to the inverse and the non-polynomial factor used in the approximation.

Further analysis of this bound reveals that

$$\frac{\max_{\xi_0, \xi_1 \in [0,1]} p(\xi_0, \xi_1)}{\min_{\xi_0, \xi_1 \in [0,1]} p(\xi_0, \xi_1)} \leq \begin{cases} 2 & \text{if } \|\boldsymbol{d}\|_2 \geq 1.094r, \\ \sqrt{2} & \text{if } \|\boldsymbol{d}\|_2 \geq 2r. \end{cases}$$

In consequence, for a shading point that is at least the sphere radius r away from the sphere surface, the most frequently sampled region may only obtain a 42% greater sample density than the most rarely sampled region. Even at a distance of $0.094r$ the sample density can only be 100% greater. In stark contrast to that, solid angle sampling has unlimited oversampling near the horizon. Our comparisons to the work of Ureña and Georgiev [2018] in Section 4.3 confirm that the resulting reduction in quality is minimal.

3.6 Sampling Cut Disks

We now introduce the efficient method for sampling cut disks that is used in all three cases. The key challenge is to find the inverse of the function $A: [-1, 1] \rightarrow [-\frac{\pi}{2}, \frac{\pi}{2}]$ (Fig. 6) given by

$$A(X) = \int_0^X 2\sqrt{1 - x^2}\,\mathrm{d}x = X\sqrt{1 - X^2} + \arcsin X.$$

Note that an inverse exists because the integrand is positive for all $x \in (-1, 1)$. We could compute the inverse numerically using Newton's method but our goal is to avoid such iterative methods. It is more efficient to compute the inverse for this single known function in one variable by crafting a specialized fit.

```
Algorithm 5 SampleCutUnitDisk
Input: An offset area $A \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and a random number $\xi_1 \in [0, 1]$.
Output: A point $d_x, d_y$ in the unit disk. The area to the left of $d_x$ is exactly $\frac{\pi}{2} + A$ and a uniform $\xi_1$ yields uniform $d_y$ across the height of the disk at $d_x$.
$g(|A|) := ((((-0.0079908617|A| + 0.0238255409)|A| - 0.0283903598)|A|$
            $+0.0198450184)|A| - 0.0574433620)|A| + 0.7400712465$
$d_x := \mathrm{signA}\left(1 - \mathrm{g(|A|)}\left(\frac{\pi}{2} - \mathrm{|A|}\right)^{\frac{2}{3}}\right)$
$d_y := (2\xi_1 - 1)\sqrt{1 - d_x^2}$
Return $d_x, d_y$
```

Fig. 6 shows the inverse function $X(A) := A^{-1}(A)$. Since it is odd, the fit only has to deal with positive inputs. The derivative of $A(X)$ (namely $2\sqrt{1 - x^2}$) is zero for $X = 1$ and thus the derivative of $X(A)$ approaches infinity as $A \longrightarrow \frac{\pi}{2}$. This poses a challenge for fits with e.g. polynomials because they can impossibly reproduce an infinite derivative. To overcome this problem, we introduce a non-polynomial factor that has a similar asymptotic behavior at $\frac{\pi}{2}$ (Fig. 6). Then our fit takes the form

$$X(A) \approx 1 - g(A)\left(\frac{\pi}{2} - A\right)^{\frac{2}{3}}$$

where $A \in [0, \frac{\pi}{2}]$ and g is a polynomial.

To determine the coefficients of the polynomial, we start from a weighted least squares fit and use it as initialization for a generic nonlinear optimizer to minimize the maximal approximation error

$$\max_{A \in [0, \frac{\pi}{2}]} \left| X(A) - \left(1 - g(A)\left(\frac{\pi}{2} - A\right)^{\frac{2}{3}}\right) \right|.$$

This optimization realizes a maximal error of $1.3 \cdot 10^{-4}$ using a cubic polynomial, $4.6 \cdot 10^{-5}$ using a quintic polynomial and $3.0 \cdot 10^{-5}$ using a sextic polynomial. We have settled for the quintic. Fig. 6 shows the resulting error distribution.

Algorithm 5 demonstrates how to use this inverse to sample a cut disk. It evaluates the polynomial g using the Horner scheme.

3.7 The Complete Sampling Algorithm

Now that the last building block is available, Algorithm 6 combines all previous algorithms into a single sampling procedure. Note that most of the work is done in all three cases such that branch divergence is reasonably low. In particular, all three sampling procedures invoke Algorithm 5 once.

For high-quality results, a single light source needs to be sampled more than once per pixel. In this case, Algorithm 6 should be split up to compute intermediate results that are independent of the random numbers ξ_0, ξ_1 only once. An implementation should also eliminate common subexpressions, which we kept in the pseudocode for the sake of readability. Algorithm 6 evaluates various square roots and the arcsine. Since the exact inputs may be arbitrarily close to the boundary of the respective domains, they should be clamped to avoid that rounding errors lead to not-a-number. Our reference implementations in HLSL and C++ include all these optimizations.

4 RESULTS

We now evaluate our technique in several ways. First, we review the sample sets with regard to their stratification. Then we describe the real-time renderer and the offline-renderer used for the

Algorithm 6 `SampleProjectedSphericalCap`
Input: Surface normal $\boldsymbol{n} \in \mathbb{S}^2$, vector to the center of a sphere $\boldsymbol{d} := \boldsymbol{q} - \boldsymbol{p} \in \mathbb{R}^3$, radius $r > 0$ and random numbers $\xi_0, \xi_1 \in [0, 1]$.
Output: A sample of the spherical cap $\boldsymbol{\omega}_i$ and the corresponding density $p(\xi_0, \xi_1)$ with respect to projected solid angle.

```
$\boldsymbol{x}, \boldsymbol{y}, z, \boldsymbol{r}, \boldsymbol{c}, \boldsymbol{t}$ := ComputeGeometry($\boldsymbol{n}, \boldsymbol{d}, r$)
If $r_x \geq 0$ and $t_x \geq 1$:
    $\boldsymbol{\omega}_i', p(\xi_0, \xi_1)$ := EntirelyAbove($\boldsymbol{r}, \boldsymbol{c}, \xi_0, \xi_1$)
Else if $r_x \geq 0$:
    $\boldsymbol{\omega}_i', p(\xi_0, \xi_1)$ := MostlyAbove($\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{t}, \xi_0, \xi_1$)
Else if $t_x < 1$:
    $\boldsymbol{\omega}_i', p(\xi_0, \xi_1)$ := MostlyBelow($\boldsymbol{r}, \boldsymbol{c}, \boldsymbol{t}, \xi_0, \xi_1$)
Else:
    The spherical cap is entirely below the horizon.
Return $\omega_{i,x}\boldsymbol{x} + \omega_{i,y}\boldsymbol{y} + \omega_{i,z}\boldsymbol{z}, p(\xi_0, \xi_1)$
```

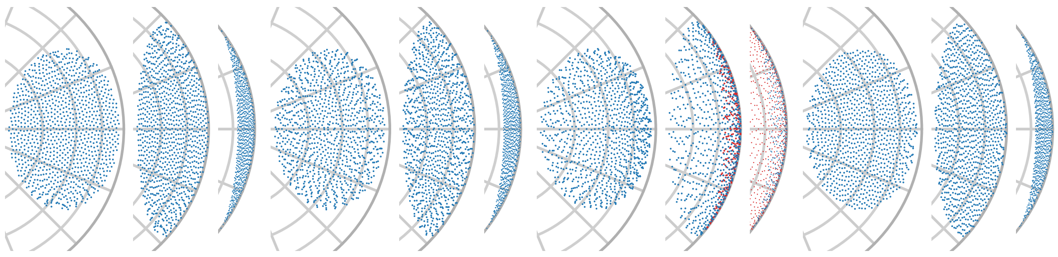

(a) Ureña and Georgiev parallel
(b) Ureña and Georgiev radial
(c) Solid angle sampling [Wang, 1992]
(d) Our projected solid angle sampling

Fig. 7. Samples produced by various techniques for the projected spherical caps in Fig. 2. The random numbers $\xi_0, \xi_1 \in [0, 1]$ are taken from a blue noise point set with 1024 points. Samples below the horizon are shown in red.

remainder of the evaluation. Finally, we discuss the quality improvement in the resulting images and provide run time measurements.

4.1 Quality of the Samples

To assess the quality of the sample sets produced by our technique and related work, we input random numbers with a high-quality blue noise spectrum[1]. Ideally, this blue noise spectrum is preserved such that samples cover the entire projected solid angle well rather than lumping together. Fig. 7 shows the results.

For all techniques, distortions are strongest where ξ_0 is close to zero or one. The parallel approach of Ureña and Georgiev [2018] uses ξ_0 to determine the y-coordinate, so distortions are strongest at the top and the bottom. With their radial approach, the largest distortions appear at the center of the ellipse and at the boundary of the projected spherical cap. Our approach has the strongest distortions in the left and right parts. When the center of the spherical cap is above the horizon,

[1]Specifically result 1 from the work of de Goes et al. [2012]: geometry.caltech.edu/BlueNoise/figs/pointsets/ours_init_ps_1024pts_1.dat

the parallel approach and our method tend to preserve the blue noise spectrum better than the radial method. For Case 3, the three techniques give similar results. Solid angle sampling produces far too many samples near and even below the horizon. Preservation of the blue noise spectrum is similar to the radial approach. The supplementary video includes an animated version of Fig. 7.

4.2 Our Renderers

Our real-time renderer uses the Falcor rendering framework [Benty et al., 2018] and Direct3D 12 with the ray tracing API DXR. It is a basic deferred renderer that shades each diffuse pixel using all spherical light sources in the scene with a fixed number of samples per light. Sampling of light sources is done with our method, solid angle sampling [Wang, 1992] or the radial approach of Ureña and Georgiev. Random numbers are taken from 256 independent 64×64 blue noise textures generated with the void-and-cluster method [Ulichney, 1993]. This approach does not provide any stratification for the random numbers used within one pixel but good stratification across pixels. For each sample, one any-hit ray is traced to determine visibility of the light source. We found that it is crucial for a good performance to unroll loops with ray tracing instructions, presumably because it allows scheduling multiple rays for a single thread at once.

The renderer uses the Disney diffuse BRDF with a roughness of 0.5 for all surfaces [Burley, 2012]. Computing the shading for a spherical light with this BRDF in closed form would be challenging, but our sampling produces an unbiased estimate with very low variance. We considered combining our technique with joint importance sampling for specular BRDFs [Dupuy et al., 2017] but in their work the authors found that simple uniform distributions lead to better approximations of specular BRDFs than clamped cosines.

We have ported the CPU implementation[2] of the radial approach of Ureña and Georgiev [2018] to HLSL for use in our real-time renderer. This version occasionally produces not-a-number because Direct3D does not support double precision for trigonometric functions and therefore we run the entire technique in single precision.

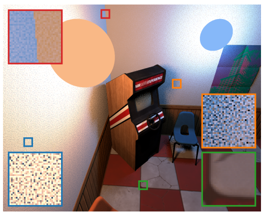
To compare to their techniques in the environment they were designed for, we additionally implemented our deferred shading pass in a simplistic offline renderer. It takes a G-buffer written by the real-time renderer and shades all pixels in the same way as the real-time renderer but without ray traced shadows. The implementation uses multithreading but no SIMD intrinsics. The techniques of Ureña and Georgiev [2018] use double precision, while our technique uses single precision. The supplementary material includes the full source code of both renderers.

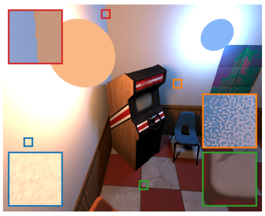
4.3 Visual Quality

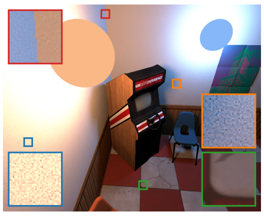
Fig. 8 shows results of our real-time renderer. The scene mostly consists of flat normal-mapped surfaces. Since the light sources are mounted to the walls, the Lambert term falls off to zero. Therefore, the variance obtained with solid angle sampling on the walls is high for the respective light source. Samples close to the horizon have a minuscule contribution. On the other hand, our projected solid angle sampling only exhibits minimal noise due to the deviations of the Disney diffuse BRDF from a constant function. Both techniques lead to noisy penumbrae but in the case of solid angle sampling this noise comes on top of the other noise. On the ground, both techniques yield fine results since it is lit from above and the Lambert term is consistently high.

Fig. 1 demonstrates the behavior of solid angle sampling and our projected solid angle sampling on curved surfaces. Naturally, any curved surface will be lit at grazing angles in some locations such that the issues of solid angle sampling become visible. These areas grow as the light source

[2]github.com/carlos-urena/psc-sampler


(a) Solid angle sampling, 1 sample


(b) Our projected solid angle sampling, 1 sample


(c) Solid angle sampling, 16 samples

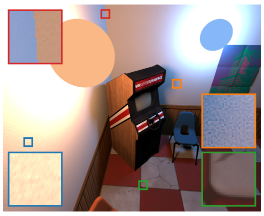
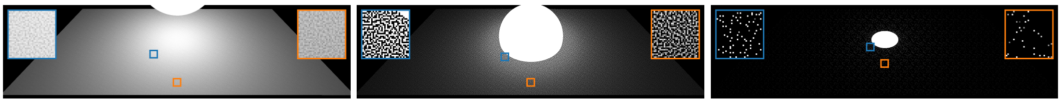
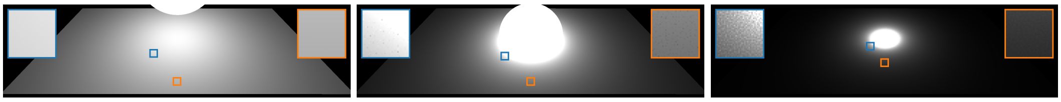
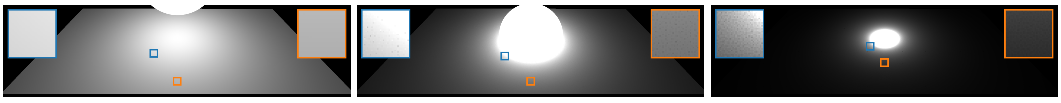

(d) Our projected solid angle sampling, 16 samples

Fig. 8. An arcade lit by two spherical light sources that are mounted to the walls. With solid angle sampling, noise is still noticeable at 16 samples per pixel per light. Results of our technique are practically noise-free at one sample per pixel per light.


(a) Solid angle sampling [Wang, 1992]


(b) Our projected solid angle sampling


(c) Ureña and Georgiev parallel

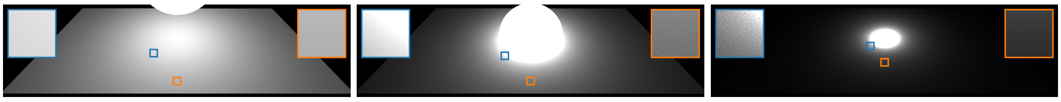

(d) Ureña and Georgiev radial

Fig. 9. A plane at $z = 0$ lit by a sphere of radius $r = 1$ with center at $q_z \in \{2, 0.1, -0.9\}$. The brightness has been adjusted for each value of q_z.

grows and they occur where spherical light sources offer more compelling shading than point lights.

Fig. 9 shows renderings of our offline renderer for a simple plane using one sample per pixel. By moving the light source up or down, we explicitly select different cases. The quality of solid angle sampling declines quickly as the light source sinks into the plane. In the extreme case $q_z = -0.9$, there are only few non-zero samples remaining. The case $q_z = -0.9$ also triggers the worst case of our sampling. Shading points next to the light source sample through Case 3. This leads to moderate noise, which is not present with the techniques of Ureña and Georgiev [2018]. As predicted, it quickly falls off as we move away from the light source. Shading points next to the light source are often overexposed such that this drawback is easy to tolerate. If the light center is above the plane,

Table 1. The cost of shading in the real-time renderer for the scene in Fig. 9. Timings are given in *picoseconds* per sample.

	1 sample per light			500 samples per light			
q_z	2.0	0.1	-0.9	2.0	0.1	-0.1	-0.9
Baseline	5.1	5.2	5.0	5.0	5.2	5.2	5.0
Solid angle sampling [Wang, 1992]	10.3	10.5	8.8	8.3	8.5	8.4	6.5
Our projected solid angle sampling	15.8	21.5	19.9	9.6	10.0	10.8	10.8
Ureña and Georgiev radial	18.6	142.6	185.9	9.1	121.3	72.2	164.1
$\frac{\text{Radial}-\text{Baseline}}{\text{Ours}-\text{Baseline}}$	1.3	8.4	12.2	0.9	24.3	11.8	27.5

all three techniques for projected solid angle sampling give very similar results. Noise is only due to deviations of the Disney diffuse BRDF from an ideal Lambertian BRDF.

The supplementary video features an example with moving light sources and demonstrates how animated noise makes the artifacts in penumbrae less noticeable.

4.4 Run Time

We measure run times on a system with an Intel Core i7-8700K having 12 logical cores and an NVIDIA RTX 2080 Ti, which features dedicated ray tracing cores. Although it is not the focus of our work, Fig. 1 includes timings, which are dominated by the cost for tracing shadow rays. The mesh has 71k triangles and 520k pixels require ray tracing. Even at 32 samples per pixel, shading only takes 4.4 ms. While this scene is simple, it shows that new hardware has brought ray traced real-time soft shadows into reach.

Table 1 lists timings of our real-time renderer with ray tracing disabled. They are full frame times divided by the total number of samples. To ensure that the arithmetic work dominates, we have used 500 light sources with one sample per pixel or one light source with 500 samples per pixel. We also measure a baseline that evaluates the Disney diffuse BRDF using a meaningless direction constructed from blue noise. The difference to timings for proper sampling techniques approximates the cost of sampling itself.

With solid angle sampling the cost decreases as the light source sinks into the plane because BRDF evaluation is skipped randomly when samples are below the horizon. Disregarding this effect, solid angle sampling takes 5.2 ps for the first sample and 3.3 ps per additional sample. Our technique becomes more expensive as it transitions from Case 1 to 2 or 2 to 3. In the worst case, it takes 14.9 ps for the first sample and 5.8 ps per additional sample. In other words, it is roughly three times more expensive than solid angle sampling with one sample per light and twice as expensive with many samples. This relative increase is significant but the absolute numbers are low. Our technique takes 0.21 ms to produce samples for $1920 \cdot 1080$ pixels with one light source and 16 samples in the worst case. The radial approach of Ureña and Georgiev [2018] performs similarly when it only has to sample an ellipse (Case 1, $q_z \geq 1$) but is 8.4 to 27.5 times slower in the other cases.

Table 2 provides timings of the offline renderer. Comparing our projected solid angle sampling to solid angle sampling [Wang, 1992] leads to similar conclusions as in the real-time renderer. Note however that our technique is roughly 800 times faster on GPU than on CPU. The parallel and radial approaches of Ureña and Georgiev [2018] perform similar to each other. For Cases 2 and 3 (i.e. $q_z < 1$), the radial approach tends to be slower but it is much faster in Case 1 because it simply samples an ellipse in the usual way. This is why the authors prefer the radial approach. In Case 1, the radial approach actually has a slightly lower cost per sample than our approach. As explained in Section 3.4, we could sacrifice the continuous behavior across cases to implement the

Table 2. The cost of shading in the offline renderer for the scene in Fig. 9. Timings are given in *nanoseconds* per sample.

	1 sample per light			500 samples per light			
q_z	2.0	0.1	-0.9	2.0	0.1	-0.1	-0.9
Baseline	2.1	2.1	2.2	2.0	2.1	2.0	2.1
Solid angle sampling [Wang, 1992]	8.2	7.4	6.3	4.8	4.0	3.8	3.1
Our projected solid angle sampling	10.1	12.9	15.1	6.3	6.4	8.4	8.4
Ureña and Georgiev radial	13.6	60.2	59.8	5.7	43.6	64.5	41.4
Ureña and Georgiev parallel	26.1	40.4	42.8	18.0	28.2	29.2	31.0
$\frac{\text{Radial}-\text{Baseline}}{\text{Ours}-\text{Baseline}}$	1.4	5.4	4.5	0.9	9.6	9.8	6.2

same fast solution but prefer not to do so for the sake of better stratification. In other cases, the radial approach is on average almost ten times slower than our technique. The cost does not only depend on the case but also on the specific value of q_z and even on the random numbers. These divergent branches lead to a greater disadvantage on the GPU in spite of the use of single-precision arithmetic.

5 CONCLUSIONS

For the time being, our technique is the only real-time technique to sample the projected solid angle of an area light. Given the great benefit for diffuse BRDFs and the negligible cost, this is a substantial step towards physically based area lights with ray traced soft shadows. Furthermore, it offers efficient shading for arbitrary diffuse BRDFs. Sampling techniques for specular BRDFs and spherical light sources are already available [Dupuy et al., 2017]. Combined with sophisticated reconstruction [Schied et al., 2017], accurate soft shadows in games and other real-time applications are in reach.

We also believe that our technique will be attractive for offline renderers. As is, we already demonstrate a significant reduction in cost with a minor reduction in quality compared to prior work [Ureña and Georgiev, 2018]. Considering that our technique is designed to run well on GPUs, it is also a good candidate to benefit from SIMD instruction sets on CPUs for further speedups.

ACKNOWLEDGMENTS

Tobias Zirr and Johannes Hanika have inspired this project through many discussions. We thank the Falcor team [Benty et al., 2018] for making GPU-accelerated ray tracing easily accessible and Ureña and Georgiev for publishing an implementation of their method. The arcade used in Fig. 8 has been modeled by Nicholas Hull at NVIDIA. The head scan of Lee Perry-Smith used in Fig. 1 is based on work at triplegangers.com. We also thank the reviewers and our colleagues for feedback on the exposition.

REFERENCES

James Arvo. 1995. Stratified Sampling of Spherical Triangles. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95).* ACM, 437–438. https://doi.org/10.1145/218380.218500

Nir Benty, Kai-Hwa Yao, Tim Foley, Conor Lavelle, and Chris Wyman. 2018. The Falcor Rendering Framework 3.2. https://github.com/NVIDIAGameWorks/Falcor

Brent Burley. 2012. Physically-based shading at Disney. https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf Technical report.

Per Christensen, Andrew Kensler, and Charlie Kilpatrick. 2018. Progressive Multi-Jittered Sample Sequences. *Computer Graphics Forum (proc. EGSR)* 37, 4 (2018). https://doi.org/10.1111/cgf.13472

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech. (proc. HPG)* 1, 2, Article 25 (Aug. 2018), 17 pages. https://doi.org/10.1145/3233305

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise Through Optimal Transport. *ACM Trans. Graph. (proc. SIGGRAPH Asia)* 31, 6, Article 171 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366190

Jonathan Dupuy, Eric Heitz, and Laurent Belcour. 2017. A Spherical Cap Preserving Parameterization for Spherical Distributions. *ACM Trans. Graph. (proc. SIGGRAPH)* 36, 4, Article 139 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073694

Manuel N. Gamito. 2016. Solid Angle Sampling of Disk and Cylinder Lights. *Computer Graphics Forum (proc. EGSR)* 35, 4 (June 2016). https://doi.org/10.1111/cgf.12946

Ibón Guillén, Carlos Ureña, Alan King, Marcos Fajardo, Iliyan Georgiev, Jorge López-Moreno, and Adrián Jarabo. 2017. Area-Preserving Parameterizations for Spherical Ellipses. *Computer Graphics Forum (proc. EGSR)* 36, 4 (June 2017). https://doi.org/10.1111/cgf.13234

Eric Heitz. 2017. Analytical calculation of the solid angle subtended by an arbitrarily positioned ellipsoid to a point source. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 852 (2017), 10 – 14. https://doi.org/10.1016/j.nima.2017.02.004

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-time Polygonal-light Shading with Linearly Transformed Cosines. *ACM Trans. Graph. (proc. SIGGRAPH)* 35, 4, Article 41 (July 2016), 8 pages. https://doi.org/10.1145/2897824.2925895

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 (short paper) (I3D '18)*. ACM, Article 2, 11 pages. https://doi.org/10.1145/3190834.3190852

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-guided Filtering: Real-time Reconstruction for Path-traced Global Illumination. In *Proceedings of High Performance Graphics (HPG '17)*. ACM, New York, NY, USA, Article 2, 12 pages. https://doi.org/10.1145/3105762.3105770

John Snyder. 1996. Area Light Sources for Real-Time Graphics. , 30 pages. https://www.microsoft.com/en-us/research/publication/area-light-sources-for-real-time-graphics/ Microsoft technical report MSR-TR-96-11.

Robert A. Ulichney. 1993. Void-and-cluster method for dither array generation. *Proc. SPIE* 1913, 1–12. https://doi.org/10.1117/12.152707

Carlos Ureña. 2000. Computation of Irradiance from Triangles by Adaptive Sampling. *Computer Graphics Forum* 2 (2000). https://doi.org/10.1111/1467-8659.00452

Carlos Ureña, Marcos Fajardo, and Alan King. 2013. An Area-Preserving Parametrization for Spherical Rectangles. *Computer Graphics Forum (proc. EGSR)* 36, 4 (June 2013). https://doi.org/10.1111/cgf.12151

Carlos Ureña and Iliyan Georgiev. 2018. Stratified Sampling of Projected Spherical Caps. *Computer Graphics Forum (proc. EGSR)* 37, 4 (2018). https://doi.org/10.1111/cgf.13471

Changyaw Wang. 1992. *Graphics Gems III*. Academic Press Professional, Chapter Physically correct direct lighting for distribution ray tracing, 307–313.

Received December 2018; revised February 2018; accepted February 2018