Eurographics Symposium on Rendering 2013
Nicolas Holzschuch and Szymon Rusinkiewicz
(Guest Editors)

Volume 32 (2013), Number 4

An Area-Preserving Parametrization for Spherical Rectangles

Carlos Ureña[1] and Marcos Fajardo[2] and Alan King[2]

[1]TARVIS Research Group, Dpt. Lenguajes y Sistemas Informáticos, Universidad de Granada, Spain.
[2]Solid Angle, Madrid, Spain.

Abstract

We present an area-preserving parametrization for spherical rectangles which is an analytical function with domain in the unit rectangle $[0,1]^2$ and range in a region included in the unit-radius sphere. The parametrization preserves areas up to a constant factor and is thus very useful in the context of rendering as it allows to map random sample point sets in $[0,1]^2$ onto the spherical rectangle. This allows for easily incorporating stratified, quasi-Monte Carlo or other sampling strategies in algorithms that compute scattering from planar rectangular emitters.

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

The map M described in this paper can be used in a number of applications which require numerical integration of functions whose domain is a *spherical rectangle*. In particular, this is the case of numerical computation of radiance directly reflected (or scattered) from a point due to illumination from a planar rectangular light source, a common task in realistic rendering systems.

The motivation for this work is two-fold. First, planar rectangles are probably the most commonly used light sources in applications such as production rendering for animation, visual effects, studio lighting and other fields. And second, the simplest and most commonly used method to sample these rectangular lights, surface area sampling, is surprisingly poor in many practical cases. More robust samplers based on combining surface area with BRDF via multiple importance sampling can do a better job but they are still far from optimal.

Our map can be used to reformulate the reflection integral as an integral in the sphere by using a change of variables which yields an integrand with lower variation. This leads to better stochastic estimators with lower variance, especially when the distance from the point to the rectangle is small or not too big relative to the size of the rectangle.

In order to design M, we will use a two-step approach similar to the one used by Arvo [Arv95] for spherical triangles, but adapted to spherical rectangles. To our knowledge, this is the first time such an analytical map is described.

2. Previous work

The need for importance sampling in direct lighting computation is well described in the literature [SW92, SWZ96]. Researchers have focused on simple planar polygons in the hope that their simplicity makes it easy to design efficient importance and/or stratified sampling strategies. The simplest technique is to use area-preserving parametrizations, or maps, from the unit square onto the planar polygon. These maps allow the implementation of importance sampling with a *probability density function* (pdf) proportional to subtended solid angle. Combined with stratified or Quasi-Monte Carlo sampling, this yields lower variance estimators for radiance.

In the case of planar triangles, Wang [Wan92] described a technique based on using area sampling on a planar triangle which is tangent to the sphere, and obtained as a projection of the original. While this contributes to smooth the integrand, for points near the source perspective distortion can be quite large and thus large errors may still arise. In his PhD thesis [Wan93], Wang stated that solid angle sampling for rectangular light sources can be done by using numerical inversion of the cumulative solid angle function. While correct, this can be quite slow for practical rendering.

The first exact analytic map for solid angle sampling of polygons was described by Arvo [Arv95]. The algorithm is quite efficient and simple to implement, and thus it is a useful improvement over naive area sampling.

Ureña described approximate adaptive cosine weighted sampling for spherical triangles [Ure00]. This technique allows to position stratified random samples in a spherical triangle, and can be easily extended to rectangles. However, the method does not provide a map but a sample generation algorithm, thus its utility is limited as it cannot be used with random, quasi-random or blue noise sample points.

Arvo described an area-preserving map for cosine-weighted sampling of arbitrary planar polygons [Arv01]. This technique is quite general, and it is based on a division of the spherical polygon in sectors between meridians through each vertex. Each sector is sampled by using a lower-order polygonal approximation for the solid angle times cosine factor. The preprocessing needed makes this approach far more complex and probably slower than the use of simple analytical functions.

3. Properties of map M

Let us consider a 3D planar rectangle P defined by one of its vertices $\mathbf{s}$ and two perpendicular vectors $\mathbf{e}_x$ and $\mathbf{e}_y$. We also consider a unit radius sphere O whose centre is a known point $\mathbf{o}$.

We define Q as the region obtained by projecting P onto O. This region is the intersection of two *spherical digons* or *lunes*. We define $\mathcal{A}$ as the area measure in a planar surface and σ as the solid-angle measure in a spherical surface.

We want to obtain a map or function M from $[0,1]^2$ to Q such that any two regions in $[0,1]^2$ with similar area map to two regions in Q with similar subtended solid angle. That is, for any region $T \subseteq [0,1]^2$, it holds that:

$$\mathcal{A}(T) \;=\; \frac{1}{\sigma(Q)}\,\sigma(M(T)) \qquad (1)$$

The function M maps any pair $(u,v) \in [0,1]^2$ to a point $\mathbf{q}$ on the spherical rectangle Q. We define $\mathbf{p}$ as the projection of $\mathbf{q}$ on the planar rectangle P in the reference system R (see Figure 1).

The point $\mathbf{q} = M(u,v)$ can be expressed in spherical coordinates by using its *colatitude* or *polar angle* β (which is a function of both u and v), and its *longitude* α (a function of u only). The cartesian coordinates of $M(u,v)$ relative to R can be expressed in terms of u and v using functions α and β:

$$\begin{aligned} M(u,v) \;\equiv\; & \mathbf{o} \;+\; \sin\alpha\sin\beta\,\mathbf{x} \;+\; \cos\beta\,\mathbf{y} \;+ \\ & \cos\alpha\sin\beta\,\mathbf{z} \end{aligned} \qquad (2)$$

where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ are the axes of local reference system R.

4. Computation of point p

In this section we will compute the coordinates of point $\mathbf{p}$ relative to R. To emphasize their dependence on u and v, the X and Y coordinates will be denoted by x_u and y_v respectively. We first need to find the solid angle of Q which is needed to compute x_u.

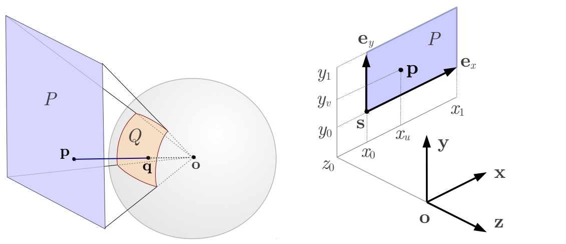


Figure 1: *Spherical rectangle Q and planar rectangle P aligned with local reference system R.*

4.1. Local reference system R

The layout of coordinate system R is critical to facilitate computing the angles and formulas for x_u and y_v. R is located at point $\mathbf{o}$ with its $\mathbf{x}$ and $\mathbf{y}$ axes aligned with the $\mathbf{e}_x$ and $\mathbf{e}_y$ edges of P. We always select $\mathbf{z}$ pointing away from P (see Figure 1). The limits of P relative to R can be obtained by simple dot products:

$$\begin{array}{lll} x_0 \equiv \mathbf{d}\cdot\mathbf{x} & y_0 \equiv \mathbf{d}\cdot\mathbf{y} & z_0 \equiv \mathbf{d}\cdot\mathbf{z} \\ x_1 \equiv x_0 + \|\mathbf{e}_x\| & y_1 \equiv y_0 + \|\mathbf{e}_y\| & \end{array}$$

where $\mathbf{d} \equiv \mathbf{s} - \mathbf{o}$. The four vertices of P relative to R are denoted by $\mathbf{v}_{00}, \mathbf{v}_{01}, \mathbf{v}_{10}$ and $\mathbf{v}_{11}$, where $\mathbf{v}_{ij} \equiv (x_i, y_j, z_0)$.

4.2. Solid angle subtended by Q

The solid angle subtended by Q can be expressed as:

$$\mathcal{A}(Q) \;=\; \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3 - 2\pi$$

where $\gamma_i = \arccos(-\mathbf{n}_i \cdot \mathbf{n}_{i\oplus 1})$. The four unit normal vectors $\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2$ and $\mathbf{n}_3$ can be obtained as the normalized cross product between the $\mathbf{v}_{ij}$ vectors:

$$\begin{array}{llllll} \mathbf{n}_0 & \equiv & \dfrac{\mathbf{v}_{00}\times\mathbf{v}_{10}}{\|\mathbf{v}_{00}\times\mathbf{v}_{10}\|} & \mathbf{n}_1 & \equiv & \dfrac{\mathbf{v}_{10}\times\mathbf{v}_{11}}{\|\mathbf{v}_{10}\times\mathbf{v}_{11}\|} \\ \mathbf{n}_2 & \equiv & \dfrac{\mathbf{v}_{11}\times\mathbf{v}_{01}}{\|\mathbf{v}_{11}\times\mathbf{v}_{01}\|} & \mathbf{n}_3 & \equiv & \dfrac{\mathbf{v}_{01}\times\mathbf{v}_{00}}{\|\mathbf{v}_{01}\times\mathbf{v}_{00}\|} \end{array}$$

These normal vectors are depicted in Figure 2. There are also four angles $\varphi_0, \varphi_1, \theta_0, \theta_1$ such that:

$$\begin{array}{lll} \mathbf{n}_0 & = & \cos\theta_0\mathbf{z} + \sin\theta_0\mathbf{y} \\ \mathbf{n}_1 & = & \cos\varphi_1\mathbf{z} + \sin\varphi_1\mathbf{x} \\ \mathbf{n}_2 & = & \cos\theta_1\mathbf{z} + \sin\theta_1\mathbf{y} \\ \mathbf{n}_3 & = & -\cos\varphi_0\mathbf{z} - \sin\varphi_0\mathbf{x} \end{array} \qquad (3)$$

4.3. Computing x_u

As in [Arv95], we use a spherical rectangle $Q_u \subseteq Q$ which is chosen in such a way that

$$\mathcal{A}(Q_u) \;=\; \mathcal{A}(Q)\,u \qquad (4)$$

Q_u is bounded by planes through $\mathbf{o}$ perpendicular to $\mathbf{n}_0$, $\mathbf{m}_u$, $\mathbf{n}_2$ and $\mathbf{n}_3$. Vector $\mathbf{m}_u$ is in the plane $y = 0$ and thus can be written as:

$$\mathbf{m}_u \;=\; \cos\varphi_u\mathbf{z} + \sin\varphi_u\mathbf{x} \qquad (5)$$

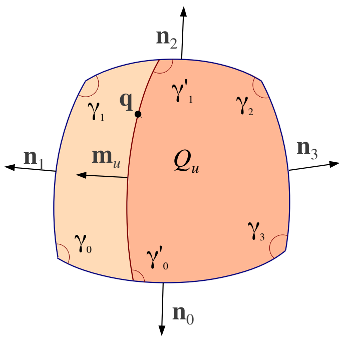


Figure 2: *Spherical rectangles Q and Q_u. As u goes from* 0 *to* 1*, $\mathbf{m}_u$ rotates from $-\mathbf{n}_3$ to $\mathbf{n}_1$ (right to left in the figure).*

Note that φ_u is an angle which depends on u, varies from φ_0 to φ_1 and determines the value of x_u. As shown in Figure 2, the internal angles of Q_u are γ_2, γ_3 (shared with Q), and γ'_0 and γ'_1 (where $\gamma'_0 \equiv \arccos(-\mathbf{n}_0 \cdot \mathbf{m}_u)$, and $\gamma'_1 \equiv \arccos(-\mathbf{n}_2 \cdot \mathbf{m}_u)$).

We want to obtain the value x_u, which depends on $\cos\varphi_u$. In order to find an expression for the latter we begin by expanding the area of Q_u:

$$\begin{aligned}\mathcal{A}(Q_u) &= \gamma'_0 + \gamma'_1 + \gamma_2 + \gamma_3 - 2\pi \\ &= \arccos(-\mathbf{n}_0 \cdot \mathbf{m}_u) + \arccos(-\mathbf{m}_u \cdot \mathbf{n}_2) \\ &\quad + \gamma_2 + \gamma_3 - 2\pi \qquad (6)\end{aligned}$$

By using (3) and (5), we can expand $\mathbf{n}_0 \cdot \mathbf{m}_u$ and $\mathbf{n}_2 \cdot \mathbf{m}_u$, obtaining:

$$\mathbf{n}_0 \cdot \mathbf{m}_u = b_0\, c_u \qquad \mathbf{n}_2 \cdot \mathbf{m}_u = b_1\, c_u \qquad (7)$$

where, for simplicity, we have used these definitions:

$$b_0 \equiv \cos\theta_0 \quad b_1 \equiv \cos\theta_1 \quad c_u \equiv \cos\varphi_u$$

From (3) we deduce that $b_0 = \mathbf{n}_0 \cdot \mathbf{z}$ and $b_1 = \mathbf{n}_2 \cdot \mathbf{z}$. Expanding equality (6) using (4) and (7), we obtain:

$$\begin{aligned}\mathcal{A}(Q)\, u &= \arccos(-b_0\, c_u) + \arccos(-b_1\, c_u) \\ &\quad + \gamma_2 + \gamma_3 - 2\pi \qquad (8)\end{aligned}$$

Appendix A shows how to express c_u as a function of u. With some additional derivations it is easy to show that:

$$x_u = -c_u \frac{z_0}{\sqrt{1-c_u^2}}$$

4.4. Computing y_v

As shown in Figure 3, point $\mathbf{p}$ is in the vertical line L defined by $x = x_u$ and $z = z_0$, thus it can be written as (x_u, y_v, z_0) where y_v is a function of v which increases from y_0 to y_1 as v goes from 0 to 1. More exactly, $\mathbf{p}$ is in the segment between $\mathbf{a}_0 \equiv (x_u, y_0, z_0)$ and $\mathbf{a}_1 \equiv (x_u, y_1, z_0)$.

In order to preserve the desired properties of M, Arvo [Arv95] showed that equal changes in v must induce through M equal changes in $\sin\beta$. This implies that, in order to find $\mathbf{q}$,

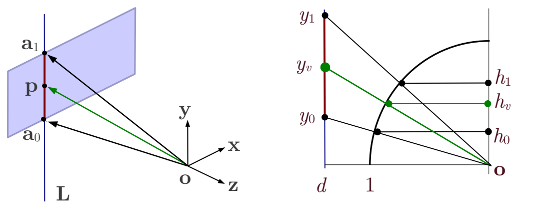


Figure 3: *Vectors $\mathbf{a}_0$ and $\mathbf{a}_1$, and line L (left), and a view of the plane spawned by $\mathbf{a}_0$ and $\mathbf{a}_1$ (right)*

we can linearly interpolate its Y coordinate (denoted by h_v) between h_0 and h_1:

$$h_v = h_0 + v\,(h_1 - h_0) \qquad (9)$$

where h_0 (resp. h_1) is the sine of the angle between $\mathbf{a}_0$ (resp. $\mathbf{a}_1$) and the plane $y = 0$:

$$h_0 \equiv \frac{y_0}{\|\mathbf{a}_0\|} = \frac{y_0}{\sqrt{d^2 + y_0^2}} \qquad h_1 \equiv \frac{y_1}{\|\mathbf{a}_1\|} = \frac{y_1}{\sqrt{d^2 + y_1^2}}$$

and where $d \equiv \sqrt{x_u^2 + z_0^2}$ is the distance from L to the Y axis. Thus y_v can be expressed as a function of h_v as follows:

$$y_v = h_v \frac{d}{\sqrt{1 - h_v^2}}$$

We now know all local coordinates of $\mathbf{p}$, which we can express in world coordinates as:

$$\mathbf{o} + x_u\,\mathbf{x} + y_v\,\mathbf{y} + z_0\,\mathbf{z}$$

5. Implementing and using map M

5.1. Algorithm overview

Appendix B provides a reference algorithm for evaluating map M which is divided in two steps. The first one stores values that are constant for all samples taken from a particular configuration of a planar rectangle and point $\mathbf{o}$ (that is, depending on the spherical rectangle). The second step utilizes the values stored in this structure as well as the values u and v to generate the point $\mathbf{p}$ on the planar rectangle. This allows an efficient implementation where the values from step 1 are precomputed and amortized over a number of calls to evaluate M.

Figure 5(c) shows the application of this algorithm to a set of 256 Hammersley points.

5.2. Sampling a PDF proportional to solid angle

Let us consider the computation of the reflected radiance at direction $\boldsymbol{\omega}_o$ at a point $\mathbf{o}$ (with normal $\mathbf{n_o}$) due to light emitted from a rectangular light source P, (with normal $\mathbf{n}_p$, pointing from P towards $\mathbf{o}$), with constant emitted radiance l. This radiance is exactly:

$$L_r(\mathbf{o}, \boldsymbol{\omega}_o) = l \int_{\mathbf{p}\in P} f_r(\mathbf{o}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, G(\mathbf{p}, \mathbf{o})\, d\mathcal{A}(\mathbf{p})$$

where the geometric term is defined as:

$$G(\mathbf{p},\mathbf{o}) \equiv \frac{\cos(\mathbf{n}_p,\boldsymbol{\omega}_i)\cos(\mathbf{n}_\mathbf{o},-\boldsymbol{\omega}_i)}{\|\mathbf{p}-\mathbf{o}\|^2}$$

and where $\boldsymbol{\omega}_i$ is the unit-length vector from $\mathbf{p}$ to $\mathbf{o}$. There are various ways of estimating this integral, such as with area sampling, solid angle sampling, or by combining either of these with other distributions using Multiple Importance Sampling (MIS) [Vea97].

Area sampling: A set of n points $\mathbf{p}_k$ are distributed in P with probability proportional to area $\mathcal{A}(P)$. This leads to the estimator:

$$L_r(\mathbf{o},\boldsymbol{\omega}_o) \approx \frac{l}{n\mathcal{A}(P)}\sum_{k=0}^{n-1} f_r(\mathbf{o},\boldsymbol{\omega}_{i_k},\boldsymbol{\omega}_o)\,G(\mathbf{p}_k,\mathbf{o})$$

Solid angle sampling: In this case, the probability measure $\mathcal{P}$ is proportional to solid angle σ, more precisely:

$$d\mathcal{P}(\mathbf{p}) = \frac{1}{\sigma(Q)}d\sigma(\mathbf{q}) = \frac{1}{\sigma(Q)}\frac{\cos(\mathbf{n}_p,\boldsymbol{\omega}_i)}{\|\mathbf{p}-\mathbf{o}\|}d\mathcal{A}(\mathbf{p})$$

Thus we arrive at the following estimator, where the full geometric term $G(\mathbf{p},\mathbf{o})$ present in area sampling has been replaced by a simpler cosine term of lower variation:

$$L_r(\mathbf{o},\boldsymbol{\omega}_o) \approx \frac{l}{n\sigma(R)}\sum_{k=0}^{n-1} f_r(\mathbf{o},\boldsymbol{\omega}_{i_k},\boldsymbol{\omega}_o)\cos(\mathbf{n}_\mathbf{o},-\boldsymbol{\omega}_{i_k})$$

Sampling probabilities Some applications (such as MIS) require the probability density for having chosen a sample point or direction through map M to be known. This probability varies with the sampling domain. We define $p_\Omega(\mathbf{q})$ as the probability for selecting point $\mathbf{q}$ in the spherical rectangle Q, while we define $p_\mathcal{A}(\mathbf{p})$ as the probability for selecting a point $\mathbf{p}$ in the rectangle P. It holds that:

$$p_\Omega(\mathbf{q}) \equiv \frac{d\mathcal{P}}{d\sigma}(\mathbf{q}) = \frac{1}{\sigma(Q)}$$

$$p_\mathcal{A}(\mathbf{p}) \equiv \frac{d\mathcal{P}}{d\mathcal{A}}(\mathbf{p}) = \frac{1}{\sigma(Q)}\frac{\cos(\mathbf{n}_p,\boldsymbol{\omega}_i)}{\|\mathbf{p}-\mathbf{o}\|^2}$$

6. Results

We have implemented this work in *Arnold*, a Monte Carlo path tracer designed for production rendering [Faj10] which makes heavy use of importance sampling techniques. Our implementation of map M provides the importance sampler an alternative sample distribution to the light-surface area-based distribution that was previously used during next-event estimation of direct lighting paths.

With regards to computational cost, sampling through M requires more complex math than the simple linear math used for generating surface area samples. And while not strictly a part of the algorithm itself, clamping of the c_u and x_u intermediate values must also be performed to avoid producing NaN or Inf values due to cumulative rounding errors in the floating point math when lighting points that are at or very near the light's tangent plane (see Listing 3). However despite these complexities, we have found that in practice there is no more than a few percent performance hit compared to area sampling in production scenes with rectangular area lights, and that this overhead can be practically eliminated in efficient implementations of the algorithm that employ fast numerical approximations to some of the math. This negligible increase to the cost of sampling is more than compensated for by the larger reduction in rays required to achieve a similar level of noise.

Figure 4 shows direct-lighting-only renderings of a simple scene with three lambertian spheres sitting on a plane illuminated by a double-sided rectangular light source that rests perpendicular to a lambertian ground. The light source has been made invisible to camera rays to better reveal areas of interest. The figure includes a false-color heatmap showing the difference of these images to a converged reference. From this heatmap it is clear that area sampling shows the largest overall difference. This is especially evident in the regions near the light source, where the division by the squared distance in the geometric term required by area sampling produces arbitrarily high sample weights.

This figure also shows how mapping M compares to the spherical triangle sampling technique described in [Arv95]. We obtain a rectangular area for this by dividing the rectangle into two triangles. To make sure the number of samples taken is the same, we apply the method of direct lighting from many luminaires described in [SWZ96] to choose one of these two triangular lights with a probability that is proportional to the contribution of each triangle in the solid angle subtended by the spherical rectangle Q. We find that despite using the same input set of samples and taking roughly the same amount of processing time, our mapping does a better job of preserving sample stratification, and thus shows slightly less noise.

Drawing samples exclusively from the surface of the light is not the only possible way of estimating direct illumination. Samples from several distributions may be combined via multiple importance sampling (MIS) to further reduce the variance of Monte Carlo estimates, sometimes dramatically [Vea97]. Figure 6 shows how combining each light sample with an additional BRDF sample using MIS can reduce variance. While using MIS results in a tremendous reduction of noise in the contact region of the light with the ground plane, it is not enough for area sampling to surpass the solid angle sampling methods, and our spherical rectangle sampling method still exhibits the least amount of noise.

Figure 7 shows the RMS error curves of these methods at different sampling rates, measured from 1-megapixel renders of our test scene. As can be seen from the chart, without MIS both of the spherical sampling methods outperform area sampling by several orders of magnitude, while using MIS reduces this enough for area sampling to only require about

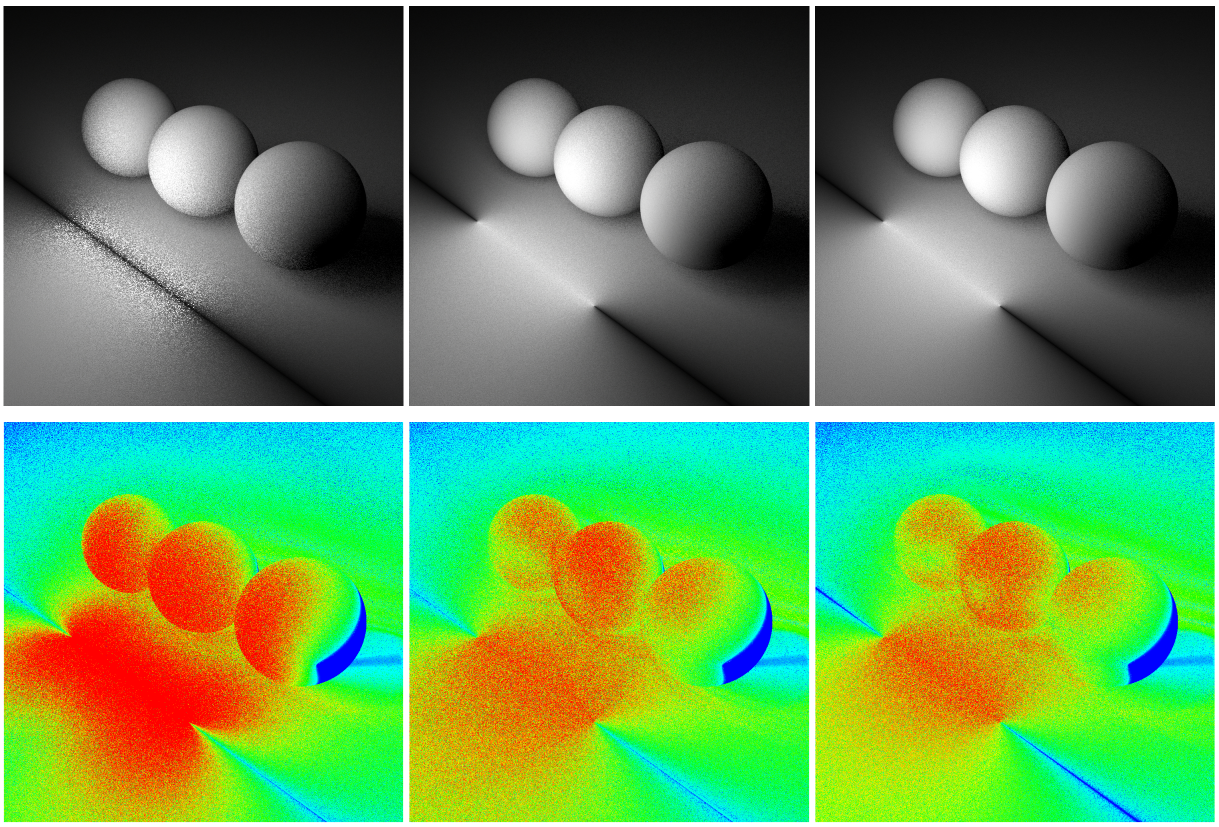


Figure 4: *First row: Double-sided rectangular light resting on a ground plane. The light has been made invisible to camera rays to better reveal areas of interest (9 paths/pixel). Second row: False colored difference to converged reference.*

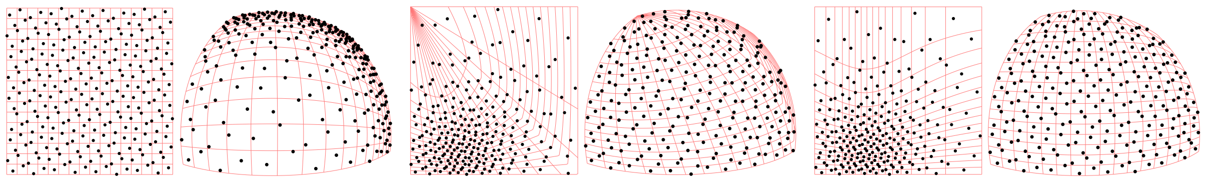


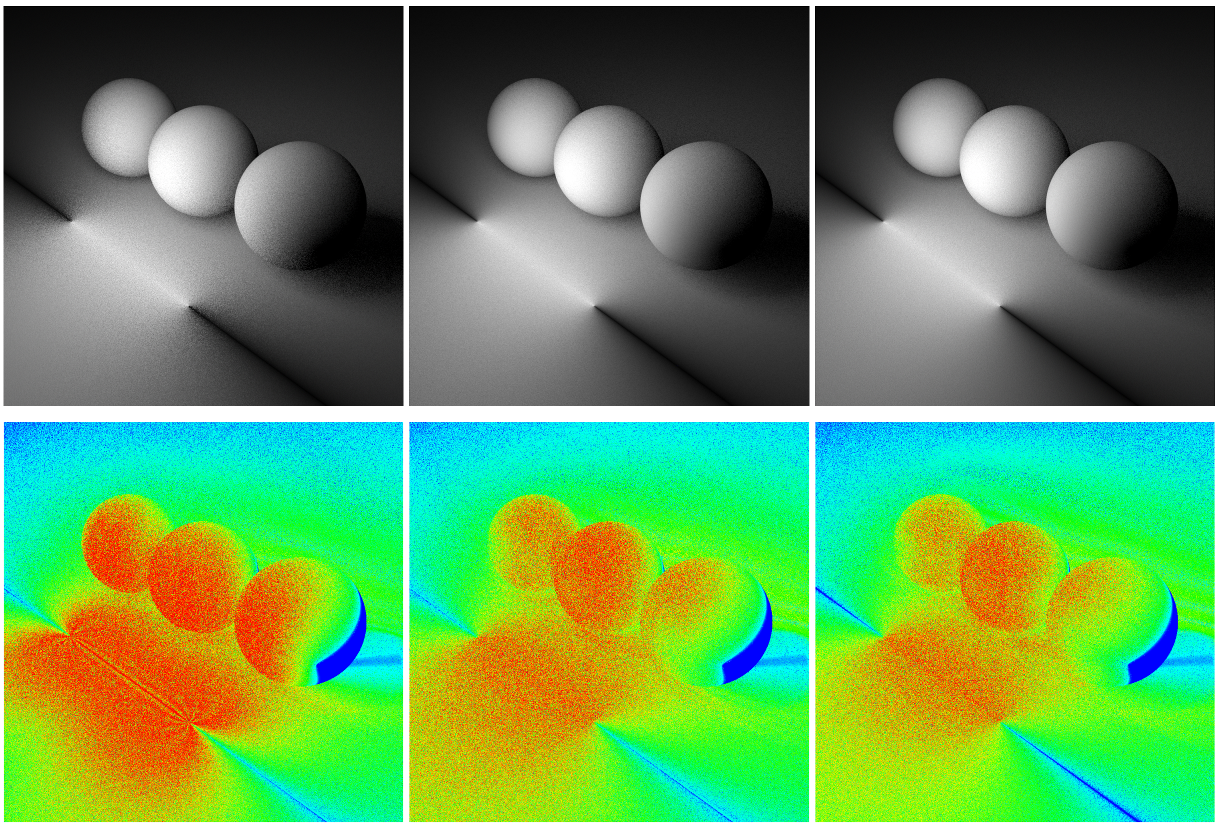
Figure 5: *A set of 256 Hammersley points mapped to a spherical rectangle using different mappings and the same points projected back onto the unit square, showing their distribution with respect to this planar area. For the solid angle-based mappings, each cell in the superimposed red grid subtends the same solid angle.*

twice as many samples as our method does to get similar levels of noise. The spherical triangle method on the other hand starts with the same error as ours at 1 sample per pixel, yet its rate of convergence is worse. The reason being that it does not preserve sample stratification as well. This allows our method to get an equal amount of error with approximately 15% less samples in our test scene when compared to spherical triangle sampling at 1024 samples per pixel, or approximately 9% less samples when MIS is used.

Figure 8 shows how our mapping performs when used to compute scattered radiance in isotropic participating media. The absence here of the surface cosine term in the illumination integrand results in a nearly optimal sampling distribution. Any remaining noise is caused by either occlusion or the one-dimensional volume integral.

7. Limitations and Future Work

While the presented mapping appears to have less variance than traditional surface area sampling of quad lights in many situations, it still has a few disadvantages. For example, it is often useful in production to place an emission texture on the surface of a light to control the detail of reflections or for image-based lighting purposes. While it is relatively straightforward to combine texture-based importance sampling with



(a) Area + BRDF (b) Two spherical triangles + BRDF (c) Spherical rectangle + BRDF

Figure 6: *First row: The same render tests from Figure 4 combined with BRDF sampling using MIS. Second row: False colored difference between images from first row and a converged reference image.*

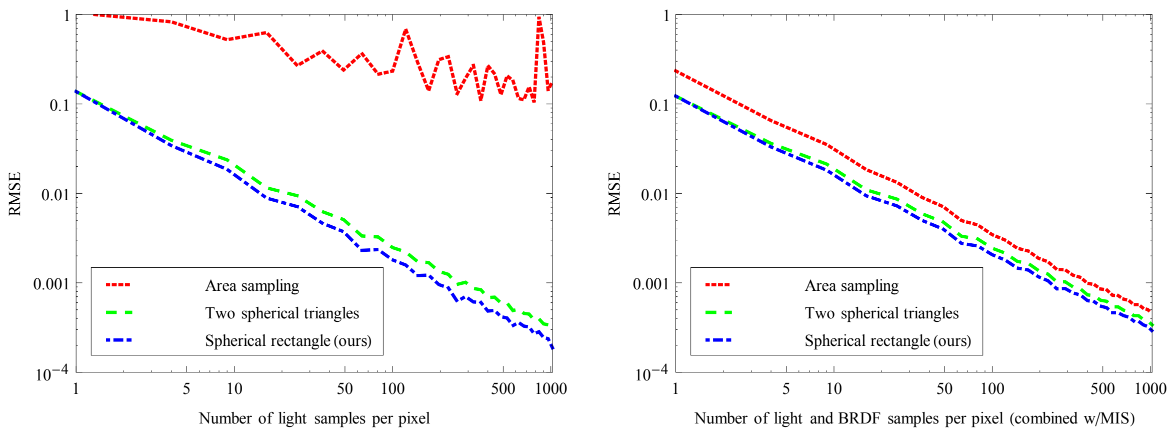


Figure 7: *Relative error curves of the different light sampling techniques on a 1 megapixel render of the diffuse spheres scene from figures 4 and 6.*

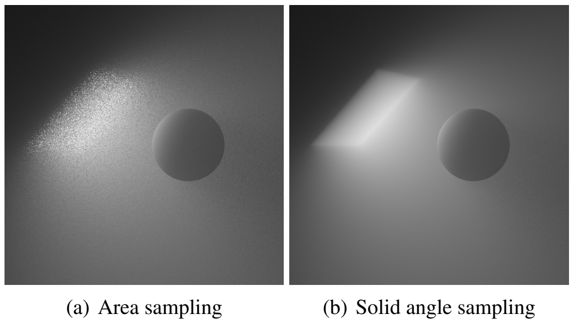


Figure 8: *Volume scattering from a single-sided rectangular light (9 paths/pixel, 2 ray march steps per ray).*

surface area sampling since the parameter spaces for the texture and irradiance samples have a trivial mapping, we have not proven whether this may be possible to do with solid angle sampling as well.

Another weakness of our method (that is shared to a different degree by area sampling) is that it does not include the cosine term present in the estimator for reflected radiance at a surface [Arv01] [Ure00]. This can result in an excess number of samples near the horizon, where they are likely to end up being less important in the final estimate. Including the cosine term in the mapping is worthy of future study.

Finally, surface area mappings exist which are capable of dealing not only with rectangles, but the larger family of planar convex quadrilaterals [AN07]. We would like to explore the feasibility of extending our technique to include convex quadrilaterals as well.

Acknowledgements

We would like to thank Solomon Boulos, Eric Haines, Thiago Ize, Chris Kulla, Anders Langlands, Emmanuel Turquin and Oscar Anson for their helpful comments in reviewing early drafts of this work. We would also like to thank the anonymous reviewers for their suggestions. Finally, we would like to dedicate this work to Jim Arvo whose pioneering ray-tracing and sampling work inspired so many of us.

References

[AN07] ARVO J., NOVINS K.: Stratified sampling of convex quadrilaterals. *Journal of Graphics, GPU, and Game Tools 12*, 2 (2007), 1–12. 7

[Arv95] ARVO J.: Stratified sampling of spherical triangles. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 437–438. 1, 2, 3, 4

[Arv01] ARVO J.: Stratified Sampling of 2-Manifolds. In *State of the Art in Monte Carlo Ray Tracing for Realistic Image Synthesis. SIGGRAPH 2001 Course Notes, volume 29* (2001), ACM. 2, 7

[Faj10] FAJARDO M.: Ray tracing solution for film production rendering. In *Global Illumination Across Industries. SIGGRAPH 2010 Course Notes* (2010), ACM. 4

[SW92] SHIRLEY P., WANG C.: Distribution ray tracing: Theory and practice. In *In Proceedings of the Third Eurographics Workshop on Rendering* (1992), pp. 33–43. 1

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics 15*, 1 (1996), 1–36. 1, 4

[Ure00] UREÑA C.: Computation of irradiance from triangles by adaptive sampling. *Computer Graphics Forum 19*, 2 (2000), 165–171. 1, 7

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. 4

[Wan92] WANG C.: Physically correct direct lighting for distribution ray tracing. 307–313. 1

[Wan93] WANG C.: *The Direct Lighting Computation in Global Illumination Methods*. PhD thesis, 1993. 1

Appendices

A. Expressing c_u as a function of u

For simplicity, we define function ϕ as:

$$\phi(x) \;\equiv\; x\mathcal{A}(Q) - \gamma_2 - \gamma_3 + 2\pi$$

Equation (8) can be rewritten as:

$$\phi(u) - \arccos(-b_0\, c_u) \;=\; \arccos(-b_1\, c_u)$$

We know that b_0, b_1 and c_u are in the range $(-1,1)$. In that range, the arccos function is a bijection, thus we can apply its inverse to both sides of the equation:

$$\cos\left(\phi(u) - \arccos(-b_0\, c_u)\right) \;=\; -b_1\, c_u$$

Applying a well known trigonometric identity, we get:

$$\begin{aligned} -b_1 c_u \;=\;& \cos\phi(u)\cos(\arccos(-b_0 c_u)) \\ &+ \sin\phi(u)\sin(\arccos(-b_0 c_u)) \end{aligned}$$

and after further simplification:

$$-\cos\phi(u)\, b_0 c_u + \sin\phi(u)\sqrt{1-b_0^2c_u^2} \;=\; -b_1 c_u \qquad (10)$$

From equations (6) and (8) we deduce that $\phi(u)$ is equal to $\gamma'_0 + \gamma'_1$. Both γ'_0 and γ'_1 are each strictly greater than $\pi/2$ and strictly smaller than π, so we conclude that $\phi(u)$ is in the open interval $(\pi, 2\pi)$, thus $\sin\phi(u) < 0$, and we can safely divide both sides in (10) by $\sin\phi(u)$:

$$\sqrt{1-b_0^2c_u^2} \;=\; \frac{\cos\phi(u)\, b_0 c_u - b_1 c_u}{\sin\phi(u)}$$

For simplicity, we define function f as:

$$f(u) \;\equiv\; \frac{\cos\phi(u)\, b_0 - b_1}{\sin\phi(u)}$$

and we get

$$\sqrt{1-b_0^2c_u^2} \;=\; c_u\, f(u)$$

In this equality, both sides are positive, therefore $\operatorname{sign}(c_u) = \operatorname{sign}(f(u))$. Squaring both sides and solving for c_u we arrive at:

$$c_u \;=\; \frac{\operatorname{sign}(f(u))}{\sqrt{f^2(u)+b_0^2}}$$

which is the expression for c_u (in terms of u) we were looking for.

B. Algorithm source code

The variable names in the C-style code below were chosen to follow the mathematical symbols in this paper as much as possible. Note that it is possible to write a more optimized algorithm by precomputing values that are constant for the planar rectangle but this is not shown here for brevity.

```
struct SphQuad {
   vec3 o, x, y, z;         // local reference system 'R'
   float z0, z0sq;             //
   float x0, y0, y0sq;         // rectangle coords in 'R'
   float x1, y1, y1sq;         //
   float b0, b1, b0sq, k;  // misc precomputed constants
   float S;                // solid angle of 'Q'
}
```

Listing 1: *Constants for the spherical rectangle Q.*

```
SphQuadInit(SphQuad& squad,vec3 s,vec3 ex,vec3 ey,vec3 o) {
   squad.o = o;
   float exl = length(ex), eyl = length(ey);
   // compute local reference system 'R'
   squad.x = ex / exl;
   squad.y = ey / eyl;
   squad.z = cross(squad.x, squad.y);
   // compute rectangle coords in local reference system
   vec3 d = s - o;
   squad.z0 = dot(d, squad.z);
   // flip 'z' to make it point against 'Q'
   if (squad.z0 > 0) {
      squad.z  *= -1;
      squad.z0 *= -1;
   }
   squad.z0sq = squad.z0 * squad.z0;
   squad.x0 = dot(d, squad.x);
   squad.y0 = dot(d, squad.y);
   squad.x1 = squad.x0 + exl;
   squad.y1 = squad.y0 + eyl;
   squad.y0sq = squad.y0 * squad.y0;
   squad.y1sq = squad.y1 * squad.y1;
   // create vectors to four vertices
   vec3 v00 = {squad.x0, squad.y0, squad.z0};
   vec3 v01 = {squad.x0, squad.y1, squad.z0};
   vec3 v10 = {squad.x1, squad.y0, squad.z0};
   vec3 v11 = {squad.x1, squad.y1, squad.z0};
   // compute normals to edges
   vec3 n0 = normalize(cross(v00, v10));
   vec3 n1 = normalize(cross(v10, v11));
   vec3 n2 = normalize(cross(v11, v01));
   vec3 n3 = normalize(cross(v01, v00));
   // compute internal angles (gamma_i)
   float g0 = acos(-dot(n0,n1));
   float g1 = acos(-dot(n1,n2));
   float g2 = acos(-dot(n2,n3));
   float g3 = acos(-dot(n3,n0));
   // compute predefined constants
   squad.b0 = n0.z;
   squad.b1 = n2.z;
   squad.b0sq = squad.b0 * squad.b0;
   squad.k = 2*PI - g2 - g3;
   // compute solid angle from internal angles
   squad.S = g0 + g1 - squad.k;
}
```

Listing 2: *Precomputation of constants for the spherical rectangle Q.*

```
vec3 SphQuadSample(SphQuad squad, float u, float v) {
   // 1. compute 'cu'
   float au = u * squad.S + squad.k;
   float fu = (cos(au) * squad.b0 - squad.b1) / sin(au);
   float cu = 1/sqrt(fu*fu + squad.b0sq) * (fu>0 ? +1 : -1);
         cu = clamp(cu, -1, 1); // avoid NaNs
   // 2. compute 'xu'
   float xu = -(cu * squad.z0) / sqrt(1 - cu*cu);
         xu = clamp(xu, squad.x0, squad.x1); // avoid Infs
   // 3. compute 'yv'
   float d  = sqrt(xu*xu + squad.z0sq);
   float h0 = squad.y0 / sqrt(d*d + squad.y0sq);
   float h1 = squad.y1 / sqrt(d*d + squad.y1sq);
   float hv = h0 + v * (h1-h0), hv2 = hv*hv;
   float yv = (hv2 < 1-eps) ? (hv*d)/sqrt(1-hv2) : squad.y1;
   // 4. transform (xu,yv,z0) to world coords
   return (squad.o + xu*squad.x + yv*squad.y + z0*squad.z);
}
```

Listing 3: *Sample function using map $M(u,v)$ that returns point* $\mathbf{p}$ *in the planar rectangle P.*