EUROGRAPHICS 2008 / G. Drettakis and R. Scopigno
(Guest Editors)

Volume 27 (2008), Number 2

Practical Product Importance Sampling for Direct Illumination

Petrik Clarberg and Tomas Akenine-Möller[†]

Lund University

Abstract

We present a practical algorithm for sampling the product of environment map lighting and surface reflectance. Our method builds on wavelet-based importance sampling, but has a number of important advantages over previous methods. Most importantly, we avoid using precomputed reflectance functions by sampling the BRDF on-the-fly. Hence, all types of materials can be handled, including anisotropic and spatially varying BRDFs, as well as procedural shaders. This also opens up for using very high resolution, uncompressed, environment maps. Our results show that this gives a significant reduction of variance compared to using lower resolution approximations. In addition, we study the wavelet product, and present a faster algorithm geared for sampling purposes. For our application, the computations are reduced to a simple quadtree-based multiplication. We build the BRDF approximation and evaluate the product in a single tree traversal, which makes the algorithm both faster and more flexible than previous methods.

Categories and Subject Descriptors (according to ACM CCS): G.3 [Probability and Statistics]: Probabilistic algorithms (including Monte Carlo) I.3.7 [Computer Graphics]: Raytracing

1. Introduction

Despite more than thirty years of research, faster and more flexible methods for solving the *rendering equation* [Kaj86] are needed to meet the demands of the industry. For high quality images, the interaction of light and materials must be accurately simulated. To obtain realistic results, the incident lighting at a real set can be captured in a high-dynamic range image, and used for lighting the scene [Deb98]. Although many different methods have been proposed, high quality rendering under environment map lighting is still a difficult problem, especially in scenes with realistic materials.

We focus on computing *direct illumination* using Monte Carlo integration, *i.e.*, the integral of the rendering equation is estimated using stochastic point sampling. The integral involves a product over incident lighting, surface reflectance, and visibility. Too few samples or a poor sampling distribution results in undesired noise. With *importance sampling*, noise is reduced by sampling important directions more densely. This is, however, difficult as some parts of the integrand are unknown.

Recently, several methods for sampling according to the product of lighting and BRDF have been proposed. Clarberg et al. [CJAMJ05] presented a general framework for wavelet-based importance sampling of products. Our algorithm is inspired by their work, but we remove most of its limitations. In *two stage importance sampling* [CETC06], heuristics are used to build a BRDF approximation per pixel. In the same spirit, we draw samples samples from the BRDF and build a hierarchical representation on-the-fly, and effectively avoid storing tabulated BRDFs. This is important, as many real world materials exhibit spatially varying reflectance (see Figure 1), and in the industry, complex procedural shaders and shading models with many parameters are common. The high dimensionality makes these materials expensive to precompute and store.

Importance sampling using hierarchical warping only requires the scaling coefficients, or *local averages*, of the product. Therefore, we first simplify the wavelet product to directly compute the product averages from the individual

[†] {petrik|tam}@cs.lth.se

wavelet coefficients. Then, we show that in our application, we can further reduce the complexity. We build the BRDF approximation hierarchically, and at the same time, compute the product by multiplying leaf nodes and propagate the results up. Both these steps are performed in a *single* tree traversal, which makes our algorithm very fast.

By sampling in world space, it is sufficient to use a single two-dimensional environment map, which is stored as a mipmap hierarchy [Wil83]. This enables very high resolution, uncompressed, lighting (*e.g.*, 4k×4k resolution), with practically no precomputation. The key contributions are:

- ⋆ We build a hierarchical approximation of the BRDF per pixel, based on a small number of point samples. Any shader that supports BRDF importance sampling can be used, as we do not rely on heuristics [CETC06].
- ⋆ A fast quadtree-based method for computing the product is presented. Compared to wavelet importance sampling [CJAMJ05], our algorithm is faster, avoids precomputation, and supports high resolution lighting.
- ⋆ We also present an optimized wavelet product, which can be useful in many other applications.
- ⋆ Our prototype implementation compares favorably to previous state-of-the-art methods, and presents a viable alternative for production rendering, where precomputation of BRDFs is infeasible.

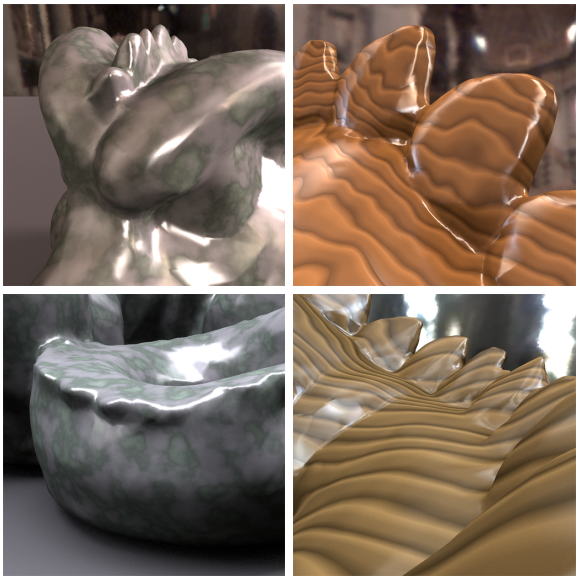


Figure 1: *Examples of procedural shaders with varying diffuse, specular, and shininess coefficients, rendered with our algorithm. Methods using precomputed BRDFs cannot easily handle these types of materials, as the high dimensionality would lead to long precomputation times and excessive memory usage. Our method samples the BRDF and builds the importance function on-the-fly, thus supporting all kinds of spatially varying materials without precomputation.*

2. Related Work

At a high level, most algorithms for photo-realistic rendering can be classified as either deterministic, stochastic, or a combination of the two. For a general overview, we refer to popular books on the subject [PH04,DBB06]. Veach [Vea97] gives an excellent overview of Monte Carlo methods for light transport problems. In the following, we limit our discussion to methods for computing the direct illumination.

Importance sampling reduces the variance by taking known information about the integrand into account to guide the sampling efforts. High-intensity regions have a larger impact on the result, and hence more samples should be placed here. There are several algorithms that sample according to only one of the involved functions, *e.g.*, environment map sampling [ARBJ03, ODJ04], and BRDF importance sampling [Shi91, War92, AS00, Mat03, CPB03, LRR04]. Work has also been done on linearly combining estimators from multiple importance functions [VG95].

Recent methods have approached the problem of sampling the *product* of lighting and surface reflectance. One approach is to first draw samples from only one of the terms, and then adjust these samples to (approximately) follow the product distribution. This can be done by *importance resampling* [BGH05, TCE05], where the initial samples are assigned weights and resampled into a smaller set, or by *rejection sampling* [BGH05], where unimportant samples are discarded. Similar to our algorithm, these methods supports spatially varying BRDFs. However, they may be inefficient when both the lighting and material have complex high-frequency features, since many samples will be useless.

In *wavelet importance sampling* [CJAMJ05], the lighting and BRDF are stored as sparse Haar wavelets, and multiplied on-the-fly using the wavelet product [NRH04]. The product is sampled by hierarchically transforming a uniform point set (*e.g.*, Halton points) into the desired distribution, using a *warping* process. The resulting samples are of high quality, but due to memory constraints, their method is limited to relatively low resolution lighting. In addition, the use of tabulated materials is a severe restriction in many applications. Cline et al. [CETC06] remove some of the limitations by hierarchically splitting the environment map based on peaks in the BRDF. The product is approximated using summed area tables, and sampled with the previously mentioned warping.

A number of other methods exist, which are not directly based on importance sampling. Ghosh and Heidrich [GH06] exploit visibility information to lower the variance. They use bidirectional importance sampling to find partially occluded pixels, and then apply Metropolis-Hastings mutations to reduce the noise in these regions. Donikian et al. [DWB*06] use adaptive importance sampling. The image is divided into small blocks, and for each block, the sampling density and a pixel estimate are updated until convergence is achieved.

The *lightcuts* framework [WFA*05,WABG06] splits the rendering integrals into sets of gather points and light points. A product traversal of these sets together with conservative termination criteria make for efficient rendering. In the same spirit, Hašan et al. [HPB07] formulate the problem as a many-light problem. They sparsely sample the transfer matrix on the GPU, and obtain impressive results.

3. Algorithmic Overview

The outgoing radiance, L_o, in a direction ω_o at a point in the scene, is given by the integral over all incident directions, ω, as follows [Kaj86]:

$$L_o(\omega_o) = \int L(\omega)\, B(\omega_o, \omega)\, V(\omega) d\omega, \tag{1}$$

where L is the incident illumination by an environment map, B is the reflectance function (see Equation 13), and V is a binary visibility term. The unbiased Monte Carlo estimator, $\hat{L}_o$, is given by:

$$\hat{L}_o = \frac{1}{N} \sum_{i=1}^{N} \frac{L(\omega_i) B(\omega_i) V(\omega_i)}{p(\omega_i)}, \tag{2}$$

where N is the number of samples, and ω_i the sampling directions. For clarity, we have omitted ω_o.

In our algorithm, we build a piecewise constant approximation, $\tilde{B}$, of the reflectance function on-the-fly, and use the product with the *exact* lighting, $L \cdot \tilde{B}$, as our importance function. Thus, the probability density function is equal to:

$$p(\omega) = \frac{L(\omega)\tilde{B}(\omega)}{L_{ns}}, \tag{3}$$

where $L_{ns} = \int L(\omega)\tilde{B}(\omega) d\omega$. The normalization by L_{ns} is necessary since, per definition, $\int p(\omega) d\omega = 1$. The value of L_{ns} is given by the root node in hierarchy of the product $L \cdot \tilde{B}$. When working with wavelets, this is equal to the first scaling coefficient of the wavelet product. Combining Equations 2 and 3, we arrive at:

$$\hat{L}_o = \frac{L_{ns}}{N} \sum_{i=1}^{N} \frac{B(\omega_i) V(\omega_i)}{\tilde{B}(\omega_i)}, \tag{4}$$

As we can see, the only remaining variance comes from the visibility, and the relative inaccuracy of our BRDF approximation, $B(\omega)/\tilde{B}(\omega)$. The main difference to wavelet importance sampling [CJAMJ05], is that we use the exact lighting instead of a wavelet approximation, $\tilde{L}$. Hence, we avoid the variance introduced by $L(\omega)/\tilde{L}(\omega)$, which can be significant with high-resolution environment maps. We also build $\tilde{B}$ on-the-fly, thereby removing the requirement of precomputed materials.

Equation 4 assumes all functions are scalar-valued. In practice, the lighting and the reflectance are usually in RGB color. We perform all computations on the three color channels, but use the *luminance* of the result as importance function. This is standard practice, but it should be noted that it adds some chrominance noise.

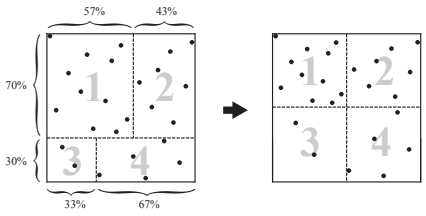


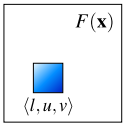
Figure 2: *The sample warping algorithm by Clarberg et al. [CJAMJ05]. A uniform point set (left) is split according to the relative intensity of each sub-quad, and the points rescaled into the desired distribution (right). When repeated hierarchically, we get a simple algorithm for sampling any importance function with a quadtree structure.*

4. Fast Product Evaluation and Sampling

In wavelet importance sampling [CJAMJ05], the individual importance functions are stored as compressed Haar wavelets, and multiplied in the wavelet domain [NRH04]. However, for importance sampling using sample warping (Figure 2), all we need are the *averages* of the quadtree nodes of the product. It is unnecessary to first compute the wavelet coefficients, and then reconstruct the averages.

This is a key observation, which we exploit to make the computations faster. We introduce two novel algorithms; an optimized wavelet product (Section 4.1) and a quadtree-based product (Section 4.2). All results were generated using the latter, so the reader may want to skim through the next section. Please refer to Appendix A and B for an introduction to wavelets and an overview of the terminology.

4.1. Fast Wavelet Product



Let $q = \langle l,u,v \rangle$ be a quad in an image $F(\mathbf{x})$, as illustrated on the right. We want to compute the *average* over q, when F is a product of two images: $F = G \cdot H$. We call this value the *parent sum* of F, or $p_{sum\,F}(q)$ for short, following the convention of Ng et al. [NRH04]. The terms p_{sum} and "average" are used interchangeably, as they represent the same thing.

The p_{sum} of a node at $\langle l,u,v \rangle$ is the sum of the coefficients of all overlapping basis functions at strictly coarser scales, $k < l$, scaled by $\pm 2^k$. The sign depends on in which quadrant of the basis function that $\langle l,u,v \rangle$ lies. We note that $p_{sum\,F}(q)$ can also be computed by integrating over $G \cdot H$ restricted to q. We define the *restricted basis*, $\Psi^{\langle q \rangle}$, of a node q as:

$$\Psi^{\langle q \rangle} = \left\{ \phi^l_{uv}, \psi_{k_1}, \ldots, \psi_{k_N} \right\}, \tag{5}$$

where k is the set of indices of all wavelet basis functions that are under the support of q, and exist at the same or finer scales. The basis $\Psi^{\langle q \rangle}$ is orthonormal due to the properties of the Haar basis, and represents a subtree of basis functions

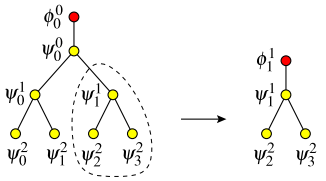


Figure 3: *On the left, a complete 1D wavelet basis tree, and on the right, a restricted basis consisting of a subtree of basis functions plus a scaling function. In two dimensions, each node has three basis functions and four children.*

with an extra scaling function appended to its root, see Figure 3. The expansion of an image $F(\mathbf{x})$ in the basis $\Psi^{\langle q\rangle}$ represents the restriction of F to q:

$$F|_q = \sum_i f_i^{\langle q\rangle}\Psi_i^{\langle q\rangle}(\mathbf{x}) = \begin{cases} F(\mathbf{x}), & \text{if } \mathbf{x}\in q,\\ 0, & \text{if } \mathbf{x}\notin q, \end{cases} \tag{6}$$

$$f^{\langle q\rangle} = \left\{2^{-l}p_{sum_F}(q), f_{k_1},\ldots,f_{k_N}\right\}, \tag{7}$$

where the scaling coefficient is equal to the node average (p_{sum}) scaled by 2^{-l}, and the detail coefficients are the same as the corresponding coefficients of the complete wavelet decomposition (Equation 16).

When F is a product of two functions, *i.e.*, $F = G\cdot H$, we can compute the average over q by integrating over the restrictions of G and H, and obtain the following:

$$\begin{aligned} p_{sum_F}(q) &= \frac{1}{A_q}\int G|_q\cdot H|_q\,d\mathbf{x}\\ &= \frac{1}{A_q}\int\left(\sum_i g_i^{\langle q\rangle}\Psi_i^{\langle q\rangle}\right)\left(\sum_j h_j^{\langle q\rangle}\Psi_j^{\langle q\rangle}\right)d\mathbf{x}\\ &= \frac{1}{A_q}\sum_i\sum_j g_i^{\langle q\rangle}h_j^{\langle q\rangle}\int\Psi_i^{\langle q\rangle}\Psi_j^{\langle q\rangle}d\mathbf{x}\\ &= \frac{1}{A_q}\sum_j g_j^{\langle q\rangle}h_j^{\langle q\rangle}, \end{aligned} \tag{8}$$

due to orthonormality. By inserting the area of q, which is $A_q = 2^{-2l}$, and the coefficients (Equation 7), we arrive at:

$$p_{sum_F}(q) = p_{sum_G}(q)\cdot p_{sum_H}(q) + 2^{2l}\underbrace{\sum_{i=1}^{N} g_{k_i}h_{k_i}}_{c_{sum_{GH}}(q)} \tag{9}$$

Here, we have introduced the notation $c_{sum_{GH}}(q)$ for the sum of the product of all coefficients in G and H, for which the wavelet functions are identical and exist under the support of q, at the same or finer scales. We call this the *children sum* of the product $G\cdot H$.

From a practical point of view, Equation 9 greatly simplifies the evaluation of the wavelet product. Given two wavelet trees, we can traverse them in parallel and precompute all c_{sum} values in a single tree traversal. As only coefficients for identical basis functions contribute, the recursion is terminated whenever a leaf node is found in one of the two trees. After computing a tree of c_{sum} values, importance sampling is reduced to hierarchically evaluating Equation 9 and warping the samples according to the intensities at each level.

4.2. Bottom-up Quadtree Product

In this section, we look at the problem of multiplying two quadtree representations, G and H. That is, the p_{sum} values are known, but not the wavelet coefficients. Our application presented in Section 5 is an important example of a case where this is useful. We build a quadtree approximation of the BRDF on-the-fly, and we wish to multiply it with an environment mipmap hierarchy, in order to sample the result.

First, we note that for all *leaf* nodes in G and H, we can compute the product average by simply multiplying the individual averages, as follows:

$$p_{sum_F}(q) = p_{sum_G}(q)\cdot p_{sum_H}(q), \tag{10}$$

where q is a leaf node in G and/or H. This follows from the fact that a quadtree leaf node is per definition constant. Hence, all its wavelet coefficients under the support of q are zero, and we can drop the $c_{sum_{GH}}$ term in Equation 9. Similarly, in all of H's children nodes, q_c, under the support of a leaf node q in G, the product is given by:

$$p_{sum_F}(q_c) = p_{sum_G}(q)\cdot p_{sum_H}(q_c), \tag{11}$$

and vice versa. Thus, we can multiply a leaf node with any other node under its support, to get the corresponding product average.

For *interior* nodes, Equation 10 does not hold. However, the product average of a node, q, can always be expressed as the average of its four immediate children nodes in the product tree, as follows:

$$p_{sum_F}(q) = \frac{1}{4}\sum_{i=1}^{4} p_{sum_F}(q+i), \tag{12}$$

where we assume $q+i$ denotes the i^{th} child of q. This leads to a simple bottom-up algorithm for computing the product quadtree. We traverse the trees of G and H in parallel, and for all leaves, we compute the product using Equation 10, and then propagate the result up using Equation 12. This can be done in a single depth-first traversal.

Once the product tree is setup, we sample it using hierarchical sample warping [CJAMJ05], starting at its root. When a leaf is reached, we proceed by evaluating Equation 11 only for the nodes where it is needed, *i.e.*, for nodes with one or more samples. This algorithm gives an efficient way to sample the product of two quadtree representations, and in addition, we completely avoid the added complexity of using wavelets. Pseudo-code and more details on our application are given in the following section.

5. Implementation

Next, we discuss the implementation of our algorithm for direct illumination under environment map lighting.

5.1. Sampling in World Space

A fundamental limitation of the wavelet product and the quadtree product, is that the functions must be defined over the same domain. Environment map lighting is given in world space, while a general BRDF is a 4D function in local space. Re-parameterization to world space gives a 7D function (or 6D for isotropic materials), which is clearly impractical. Clarberg et al. [CJAMJ05] solve the problem by pre-rotating the lighting into local space for a dense set of directions. This limits their algorithm to low resolution lighting, *e.g.*, 128^2 or 256^2 ($\sim$1GB compressed).

Using a low resolution approximation of the lighting to guide the sampling, introduces a substantial amount of noise when a higher resolution environment map is used, as we will see in Section 6. For realistic high-frequency lighting, environment maps of 1k$\times$1k to 4k$\times$4k resolution are common. As pre-rotation of such large maps is currently infeasible due to memory usage and precomputation time, importance sampling must be performed directly in world space. This also requires the BRDF to be in world space.

One option is to rotate a 2D slice of the BRDF into world space on-the-fly using wavelet rotation matrices [WNLH06]. This can be made fast enough by exploiting the sparsity of the matrices and the BRDF, but the memory is still a limiting factor. For example, with 64^2 distinct rotations, and a source and target resolution of 64^2, the rotation matrices require about 2GB. We have tried this approach, but the results were satisfactory only for diffuse materials. For glossy BRDFs, the misalignment of the specular peak due to the discretization introduces a large amount of noise.

The solution we settled for, is instead to build a BRDF approximation in world space on-the-fly, based on a small number of point samples. The approximation is multiplied by the environment map using the fast quadtree-based product, and the result is sampled using warping.

5.2. Environment Map

The environment map is stored as a single uncompressed high resolution image, together with its mipmap hierarchy. Each pixel in the hierarchy stores the average over its four immediate children pixels. With the quadtree-based product (Section 4.2), we do not need to store the wavelet coefficients. Hence, the memory requirement is only 33% larger than the environment map itself. As we do not rely on tabulated materials, the total setup time is reduced to computing a single mipmap hierarchy. For a 4k$\times$4k map, this takes less than one second.

To represent directions on the sphere, we use a mapping from the sphere to a *single* quad, which is based on the octahedral map [PH03], but with an area-preserving parameterization. The details are given in Appendix C.

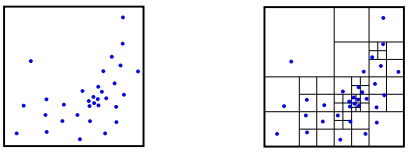


Figure 4: *Based on a set of point samples mapped onto the unit square (left), we build a quadtree approximation of the reflectance function by recursively subdividing the set until only one sample per node remains (right).*

5.3. BRDF Approximation

For a specific viewing direction, ω_o, the *bidirectional reflectance distribution function* (BRDF) is a two-dimensional function over all incident directions, ω. We define the local *reflectance* function, B, as the BRDF, f_r, times the cosine term in world space, as follows:

$$B(\omega) = \begin{cases} f_r(\omega_o, \omega)(\omega \cdot n), & \omega \cdot n > 0, \\ 0, & \omega \cdot n \leq 0, \end{cases} \tag{13}$$

where n is the surface normal. Assume we have a set of point samples, $S = \{B(\omega_1), \ldots, B(\omega_N)\}$, taken from the reflectance function. In the next section, we will describe how S is chosen. The problem is to reconstruct a continuous surface, $\tilde{B}$, which is a reasonable approximation to B. This is a scattered data interpolation problem, and ideally, we would like to use higher-order techniques. However, in our framework, we are limited to a piecewise constant quadtree approximation.

We build a quadtree by recursively dividing the set of samples, S, until only one sample per node remains. This is illustrated in Figure 4. Each internal node stores the average (*brdf_psum*) of its four children, and empty leaf nodes are assigned the value of their nearest parent. We also augment our quadtree nodes with a field storing the product average (*prod_psum*) over the node, as follows:

struct node
  brdf_psum : BRDF parent sum (average)
  prod_psum : product parent sum (average)
  ch[4] : children pointers
end

Pseudo-code for building the BRDF approximation and computing the product tree is given in Algorithm 1. Input to the algorithm is the set of all BRDF samples, $S = \{B(\omega_i)\}$, and the root node of the tree, located at $q = \langle 0,0,0 \rangle$.

5.4. Obtaining the BRDF Samples

The quality of $\tilde{B}$ naturally depends on how the samples, S, are chosen. Dense sampling results in a finer subdivision,

```
1  function BUILDPRODUCT(quad q, node n, samples S)
2      if size(S) = 1 then
3          n.brdf_psum = S[1].value
4          n.prod_psum = S[1].value × L.psum(q)
5      else
6          n.ch[1..4] = new node()      // initialized to null
7          Split S into bin[1..4] based on S[i].position
8          j = indices of non-empty bins
9          for i ∈ j
10             BUILDPRODUCT(q+i, n.ch[i], bin[i])
11         n.brdf_psum = avg( n.ch[j].brdf_psum )
12         for i ∉ j
13             n.ch[i].brdf_psum = n.brdf_psum
14             n.ch[i].prod_psum = n.brdf_psum ×
15                                 L.psum(q+i)
16         n.prod_psum = avg( n.ch[1..4].prod_psum )
```

Algorithm 1: *Recursive function for building the approximation, $\tilde{B}$, based on a set of point samples, S, while simultaneously computing all product averages in a single tree traversal. The mipmap hierarchy of the environment map is denoted L, and the avg() function computes the average of the supplied values. The current node is identified by q, and $q+i$ is assumed to identify the i^{th} child of q.*

and the area of each leaf node is approximately proportional to the inverse of the local sampling density. We assume the samples are drawn from some probability density $p_B(\omega)$.

Uniform sampling works well for diffuse materials. However, for more specular BRDFs, we would likely miss high-frequency features. Therefore, we adopt an importance sampling strategy. The error in the reconstruction, ε, is equal to the difference between the original and the approximated function. Since $\tilde{B}$ will be used as an importance function, it is desirable to distribute the approximation error evenly. It is likely that ε is larger in high-intensity regions than in regions with low intensity. Hence, ideally, we want the sampling density to be proportional to the reflectance function, *i.e.*, $p_B(\omega) \propto B(\omega)$. This way, we get higher precision around important features, such as bright specular peaks, and less resolution in smoother regions.

However, for many analytical reflectance models, sampling according to the BRDF times the cosine term is non-trivial. We often have to choose $p_B(\omega) \propto f_r(\omega_o, \omega)$. The exact choice of sampling density is not critical as it does not affect the correctness of our algorithm, but only the quality of the resulting importance function. As the sampling is done on-the-fly, it is in many cases preferable to choose a slightly inferior, but faster, sampling strategy.

Some examples of BRDFs for which analytical sampling is well known include the modified Phong model [LW94], and the anisotropic Ward and Ashikhmin models [War92, AS00] among others. Many of these sampling strategies are already implemented in existing rendering packages. This is a great advantage, as it makes the implementation of our algorithm a relatively easy task.

For measured materials, a number of sampling methods exist. We can use, for example, wavelet-based methods [Lal97, Mat03, CPB03] or factorization [LRR04]. Also note that the BRDF sampling step can be precomputed for all materials that are not spatially varying. We discretize the outgoing direction, and for each direction, a set of samples in local space is computed. At runtime, the samples are rotated into world space, which is very fast. We have found that a few hundred point samples is enough to build a quadtree approximation with a quality equivalent to, or better than, wavelet-compressed tabulated BRDFs [CJAMJ05].

```
1  function WARPPRODUCT(points P, quad q, node n)
2      if q.level = max_level then
3          Compute probability density for each P[i]
4          Store P in sample array
5          return
6
7      for i = 1 to 4
8          if is_leaf(n) then
9              w[i] = intensity(n.brdf_psum×L.psum(q+i))
10         else
11             w[i] = intensity( n.ch[i].prod_psum )
12     Compute splitting planes based on w[1..4]
13     Warp P into bin[1..4]
14
15     for i = 1 to 4
16         if size(bin[i]) > 0 then
17             if is_leaf(n) then
18                 WARPPRODUCT(bin[i], q+i, n)
19             else
20                 WARPPRODUCT(bin[i], q+i, n.ch[i])
21     return
```

Algorithm 2: *Recursive algorithm for warping an initially uniform point set, P, according to the product quadtree. When a leaf node is reached, the sampling continues according to the environment map, L, up to its full resolution. The intensity() function computes the luminance of an RGB color.*

5.5. Product Sampling

After computing the product quadtree (Algorithm 1), we proceed by sampling it using sample warping [CJAMJ05]. At each level, a horizontal and two vertical splitting planes are computed based on the product averages of the four immediate children, and the samples are rescaled accordingly, as illustrated in Figure 2.

When a leaf node in the product tree is reached, the sampling continues according to the environment mipmap hierarchy. Thus, although the BRDF approximation is of limited resolution, the sample warping is not terminated until the full resolution of the environment map is reached. Pseudo-code is given in Algorithm 2.

6. Results

We have implemented the algorithm described in Section 5 in a custom ray tracer loosely based on `pbrt` [PH04]. All images are unbiased and were rendered on a MacBook Pro with Intel Core 2 Duo 2.40GHz (using only *one* core). The current implementation does not use any SIMD-optimizations. However, this would be fairly straightforward to add, since many of our operations are performed on four children nodes in parallel. For all tests, the BRDFs were sampled on-the-fly, but it should be noted that precomputed samples (Section 5.4) can be used to further improve the performance in many cases.

Low-discrepancy points with good spectral properties are essential for lowering the variance in any Monte Carlo technique. We use the method of Dunbar and Humphreys [DH06] to quickly generate Poisson-disk points that are fed to the sampling algorithm.

Figure 7 compares our algorithm with two state-of-the-art methods for product importance sampling: *wavelet importance sampling* (WIS) by Clarberg et al. [CJAMJ05], using the unbiased version of their algorithm, and Cline et al.'s [CETC06] *two stage importance sampling*. Both these methods are based on sampling the product of lighting and BRDF. The scene is lit by a 1k×1k light probe featuring a small strong light source: the sun. The dragon uses a Phong shader, and the ground plane is purely diffuse. For sampling the BRDFs, 256 point samples each were used for the diffuse and specular lobes. The rendering times at 1024×768 pixels resolution, using one primary ray per pixel and varying number of visibility samples, were (*min:sec*):

#samples	Cline et al.	Clarberg et al.	Our algorithm
10	1:00	2:40	1:39
30	2:22	3:16	2:15
100	7:22	5:12	4:09
300	19:13	10:15	9:14

This scene presents a major challenge for WIS, which is limited to a low resolution wavelet approximation of the lighting (*e.g.*, 128^2 or 256^2). The error introduced by the approximation, *i.e.*, $L(\omega)/\tilde{L}(\omega)$, significantly increases the variance, especially in unoccluded regions. In addition, their method shows banding in noisy regions, which comes from the varying accuracy of the importance function due to bilinear interpolation of the wavelet terms. We completely avoid these problems by sampling in world space.

Two stage importance sampling handles this scene much better, and similar to our algorithm, it supports spatially varying reflectance functions. However, for equal variance, our algorithm gives a $1.5\times-2.7\times$ reduction in the number of visibility samples, as shown by the plot in Figure 7. It should be noted that a direct comparison of the results is difficult, as the rendering systems differ slightly and the exact speed depends on what type of shaders are used. The reported timings suggest that our ray tracer is faster than Cline et al.'s, although the acceleration data structure is the same (`pbrt`). Figure 5 shows an equal-time comparison for a simple scene lit by the "kitchen" light probe.

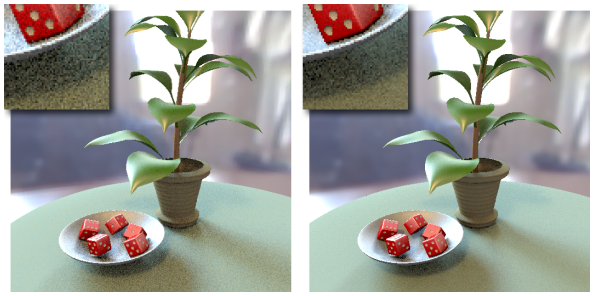


Figure 5: *Two images rendered in equal time (28 seconds) using two stage importance sampling (left) [CETC06], and our algorithm (right). The noise is significantly reduced.*

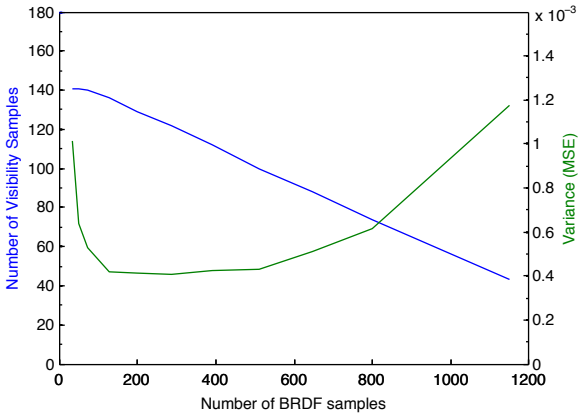


Figure 6: *Equal-time comparison for different sample allocations using the scene in Figure 7. The green line shows the variance for different numbers of BRDF samples (x-axis) and visibility samples (blue line). In this example, the best results are obtained with approximately 100–500 BRDF samples. This shows that our algorithm is robust with respect to the choice of sampling rates, and a reasonable default value (e.g., 256 BRDF samples) works well in most cases.*

Our algorithm is essentially a two step method. First, the shader is sampled, and then the product is computed and sampled. An important consideration is the allocation of samples between the two steps. In Figure 6, we have varied the number of BRDF samples (specular+diffuse), while keeping the rendering time constant by adjusting the number of visibility samples. The optimal ratio is, of course, determined by the relative speed of BRDF evaluations versus ray tracing. In our implementation, peak performance is reached with about 100–500 BRDF samples. Interestingly, Figure 6 also shows that the cost of constructing and sampling the importance function grows only linearly with the number of point samples used for approximating the material.

7. Discussion

We have presented several practical improvements to wavelet importance sampling [CJAMJ05]. To avoid the limitations of tabulated materials, we build a BRDF approximation on-the-fly. We also replace the wavelet product with a simpler quadtree-based product, computed in a single traversal, and thus effectively avoid wavelets altogether. The precomputation is reduced to a creation of a mipmap hierarchy for the lighting, and the memory requirements are very modest.

The proposed algorithm has much in common with Cline et al.'s method [CETC06]. The main differences are the evaluation of the product (quadtree *vs* summed area table) and the construction of the BRDF approximation. We rely on BRDF importance sampling, while Cline et al. use heuristics specifically designed for each supported BRDF. Their approach does not require importance sampling of the shader, which is a big advantage, but finding good heuristics for general materials can be tedious. On the other hand, our algorithm may be difficult to use with some complex shaders, for which none or only poor sampling strategies exist.

The two methods produce comparable results, although the noise in occluded regions is lower with our algorithm. This indicates that our quadtree-based BRDF approximation is slightly more accurate. In production rendering, the main bottleneck is currently the shading. Methods for creating good BRDF approximations based on a minimal number of shader evaluations are needed. We hope our work will stimulate research in that direction.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback, Jacob Munkberg for the scene in Figure 5, and the people in our lab for their support. This work was supported by the Swedish Foundation for Strategic Research and by Intel Corporation.

References

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured Importance Sampling of Environment Maps. *ACM Transactions on Graphics, 22*, 3 (2003), 605–612.

[AS00] ASHIKHMIN M., SHIRLEY P.: An Anisotropic Phong BRDF Model. *Journal of Graphics Tools, 5*, 2 (2000), 25–32.

[BBS94] BERMAN D. F., BARTELL J. T., SALESIN D. H.: Multiresolution Painting and Compositing. In *Proceedings of ACM SIGGRAPH* (1994), pp. 85–90.

[BGH05] BURKE D., GHOSH A., HEIDRICH W.: Bidirectional Importance Sampling for Direct Illumination. In *Eurographics Symposium on Rendering* (2005), pp. 147–156.

[CETC06] CLINE D., EGBERT P. K., TALBOT J. F., CARDON D. L.: Two Stage Importance Sampling for Direct Lighting. In *Eurographics Symposium on Rendering* (2006), pp. 103–113.

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM Transactions on Graphics, 24*, 3 (2005), 1166–1175.

[CPB03] CLAUSTRES L., PAULIN M., BOUCHER Y.: BRDF Measurement Modelling using Wavelets for Efficient Path Tracing. *Computer Graphics Forum, 22*, 4 (2003), 701–716.

[DBB06] DUTRÉ P., BEKAERT P., BALA K.: *Advanced Global Illumination*, second ed. A K Peters, 2006.

[Deb98] DEBEVEC P.: Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of ACM SIGGRAPH* (1998), pp. 189–198.

[DH06] DUNBAR D., HUMPHREYS G.: A Spatial Data Structure for Fast Poisson-disk Sample Generation. *ACM Transactions on Graphics, 25*, 3 (2006), 503–508.

[DWB*06] DONIKIAN M., WALTER B., BALA K., FERNANDEZ S., GREENBERG D. P.: Accurate Direct Illumination Using Iterative Adaptive Sampling. *IEEE Transactions on Visualization and Computer Graphics, 12*, 3 (2006), 353–364.

[GH06] GHOSH A., HEIDRICH W.: Correlated Visibility Sampling for Direct Illumination. *The Visual Computer, 22*, 9 (2006), 693–701.

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix Row-Column Sampling for the Many-Light Problem. *ACM Transactions on Graphics, 26*, 3 (2007), 26.

[Kaj86] KAJIYA J. T.: The Rendering Equation. *Computer Graphics (Proceedings of ACM SIGGRAPH), 20*, 4 (1986), 143–150.

[Lal97] LALONDE P.: *Representations and Uses of Light Distribution Functions*. PhD thesis, University of British Columbia, 1997.

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient BRDF Importance Sampling using a Factored Representation. *ACM Transactions on Graphics, 23*, 3 (2004), 496–505.

[LW94] LAFORTUNE E. P., WILLEMS Y. D.: *Using the Modified Phong BRDF for Physically Based Rendering*. Tech. Rep. CW197, Katholieke Universiteit Leuven, 1994.

[Mat03] MATUSIK W.: *A Data-Driven Reflectance Model*. PhD thesis, MIT, 2003.

[NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics, 23*, 3 (2004), 477–487.

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast Hierarchical Importance Sampling with Blue Noise Properties. *ACM Transactions on Graphics, 23*, 3 (2004), 488–495.

[PH03] PRAUN E., HOPPE H.: Spherical Parametrization and Remeshing. *ACM Transactions on Graphics, 22*, 3 (2003), 340–349.

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[SC97] SHIRLEY P., CHIU K.: A Low Distortion Map between Disk and Square. *Journal of Graphics Tools, 2*, 3 (1997), 45–52.

[SDS96] STOLLNITZ E. J., DEROSE T. D., SALESIN D. H.: *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, 1996.

[Shi91] SHIRLEY P. S.: *Physically Based Lighting Calculations*

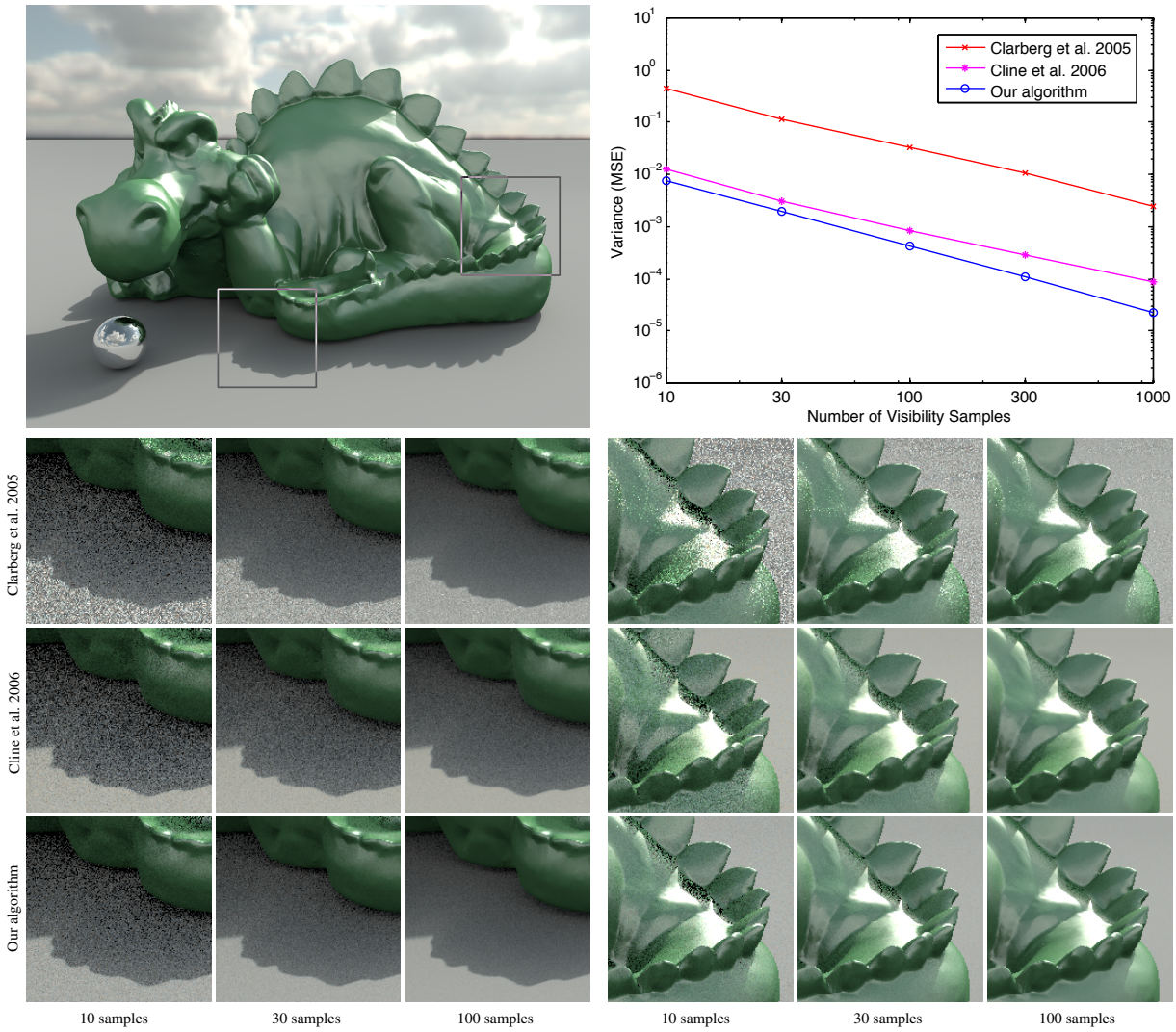


Figure 7: *We compare our method (bottom row) with wavelet importance sampling (top row) [CJAMJ05], and two stage importance sampling (middle row) [CETC06], using 10, 30, and 100 visibility samples/pixel. For this test, we used 256 BRDF samples for the diffuse floor, and 512 BRDF samples for the glossy green material. All images are unbiased, and ground truth is shown on the top left. Clarberg et al.'s method is noisy in unoccluded areas, as their low-resolution lighting approximation fails to capture the precise location of the small bright light (the sun). Cline et al.'s method gives better results, but exhibits more noise than our algorithm in shadow regions due to their more crudely approximated importance functions. For equal variance, our method gives a 1.5×−2.7× reduction in the number of visibility samples compared to their method. The variance was measured on the rendered HDR images before tone-mapping. The light probe is courtesy of Paul Debevec.*

for Computer Graphics. PhD thesis, University of Illinois at Urbana-Champaign, 1991.

[SM06] SUN W., MUKHERJEE A.: Generalized Wavelet Product Integral for Rendering Dynamic Glossy Objects. *ACM Transactions on Graphics, 25*, 3 (2006), 955–966.

[TCE05] TALBOT J., CLINE D., EGBERT P.: Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering* (2005), pp. 139–146.

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

[VG95] VEACH E., GUIBAS L. J.: Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of ACM SIGGRAPH* (1995), pp. 419–428.

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional Lightcuts. *ACM Transactions on Graphics, 25*, 3 (2006), 1081–1088.

[War92] WARD G. J.: Measuring and Modeling Anisotropic Reflection. *Computer Graphics (Proceedings of ACM SIGGRAPH),*

26, 2 (1992), 265–272.

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A Scalable Approach to Illumination. *ACM Transactions on Graphics, 24*, 3 (2005), 1098–1107.

[Wil83] WILLIAMS L.: Pyramidal Parametrics. *Computer Graphics (Proceedings of ACM SIGGRAPH), 17*, 3 (1983), 1–11.

[WNLH06] WANG R., NG R., LUEBKE D., HUMPHREYS G.: Efficient Wavelet Rotation for Environment Map Rendering. In *Eurographics Symposium on Rendering* (2006), pp. 173–182.

Appendix A: Wavelet Primer: The 2D Haar Basis

The two-dimensional (nonstandard) Haar basis is an orthonormal basis made up of translations and dilations of mother basis functions defined over the unit square. The normalized *scaling basis functions* and *wavelet basis functions* are defined as [SDS96]:

$$\phi^l_{uv}(x,y) = 2^l\phi(2^l x-u, 2^l y-v), \quad (14)$$

$$\psi_M{}^l_{uv}(x,y) = 2^l\psi_M(2^l x-u, 2^l y-v), \quad (15)$$

where u,v are integer translations in $[0,2^l-1]$, and l is a positive integer representing the *scale*, which goes from coarse to fine. The mother wavelet functions, ψ_M, are defined in Figure 8, and the mother scaling function is $\phi(x,y)=1$ for $x,y\in[0,1]^2$, and 0 elsewhere. A two-dimensional image, F, with $2^k\times 2^k$ elements can be exactly represented in the basis consisting of the first scaling function and all wavelet functions up to scale $k-1$. For convenience, we denote this basis $\Psi=\{\phi_0,\psi_1,\ldots,\psi_N\}$, where $\phi_0=\phi^0_{0,0}$ and ψ_j are the wavelet functions, sequentially numbered. The expansion of F can be written:

$$F=\sum_{i=0}^{N} f_i\Psi_i, \quad (16)$$

where the wavelet coefficients, f_i, are given by the inner product: $f_i=\int F(x,y)\Psi_i(x,y)dxdy$. We call f_0 the *scaling* coefficient, and the rest *detail* coefficients.

Appendix B: Quadtree Encoding and Wavelet Products

The scale and translation of a basis function uniquely identifies the *wavelet square* in which it resides. We use $\langle l,u,v\rangle$ to denote a square at scale l and offset u,v. All squares at the same level are disjoint, and each has an area of $A=2^{-2l}$. Since a square has four children, it is natural to encode 2D wavelet coefficients in a *quadtree* structure [BBS94, SM06]. With sparse wavelet representations, one or more of the coefficients and/or children of a node may be missing. Hence, empty interior nodes are possible.

The product of two 2D images, $F=G\cdot H$, can be efficiently computed directly in the wavelet basis [NRH04]. The theory has later been generalized to higher dimensions [CJAMJ05], and to include multiple terms [SM06]. The quadtree structure of the Haar basis is essential for reducing the cost. Sun and Mukherjee [SM06] showed that the product can be described as directed paths through the tree of basis functions. In the same spirit, we exploit the structure to optimize the sampling of wavelet products.

Figure 8: *The two-dimensional Haar mother wavelet basis functions, ψ_M, are defined over the unit square with the values $+1$ where red, -1 where blue, and 0 elsewhere.*

Appendix C: Area-Preserving Mapping of the Sphere

To simplify the sampling, we need an area-preserving mapping that, ideally, maps a single square to the sphere, with low distortion and fast analytical forward and inverse transforms. We combine the octahedral map [PH03] with the parametrization of Shirley and Chiu [SC97] to obtain a mapping with all the desired properties, as illustrated in Figure 9. As the mapping is an important practical aspect of our implementation, we repeat the formulas here.

The "inner" quad (rotated by 45°) maps to the northern hemisphere, while the outer four triangles are folded down to cover the southern hemisphere. Shirley and Chiu map a square to the unit disk, and then to the hemisphere, to obtain an area-preserving mapping. Figure 10 illustrates the square-to-disk mapping for the inner triangle of the first quadrant. Given a point (u,v) in the triangle, the lengths of a and b are $a=(u+v)/\sqrt{2}$ and $b=\sqrt{2}v$. The mapping to the disk is:

$$r = \sqrt{2}a = u+v,$$
$$\phi = \frac{\pi}{4}\frac{b}{a} = \frac{\pi}{2}\frac{v}{u+v}. \quad (17)$$

Similar transforms apply to the other quadrants. The point (r,ϕ) is then projected onto the northern hemisphere, while preserving fractional area, as follows [SC97]:

$$(x,y,z) = (r\sqrt{2-r^2}\cos\phi,\ r\sqrt{2-r^2}\sin\phi,\ 1-r^2). \quad (18)$$

This mapping uses the same number of trigonometric operations as the cylindrical equal-area projection, but the distortion is much more well-behaved. The inverse transform is well-defined and fast to compute. Interpolation across the seams is also easy due to the boundary symmetry of the octahedral map [PH03].

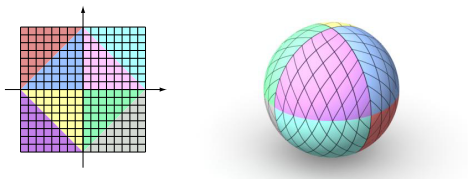


Figure 9: *The square is divided into $n\times n$ pixels, which are mapped to the same number of irregularly shaped quads with equal area on the sphere.*

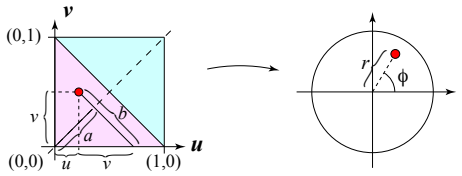


Figure 10: *The mapping of the first quadrant to the disk.*