Eurographics Symposium on Rendering (2006)
Tomas Akenine-Möller and Wolfgang Heidrich (Editors)

Two Stage Importance Sampling for Direct Lighting

David Cline[1], Parris K. Egbert[1], Justin F. Talbot and David L. Cardon[1]

[1]Brigham Young University, Provo Utah, USA

Abstract

We describe an importance sampling method to generate samples based on the product of a BRDF and an environment map or large light source. The method works by creating a hierarchical partition of the light source based on the BRDF function for each primary (eye) ray in a ray tracer. This partition, along with a summed area table of the light source, form an approximation to the product function that is suitable for importance sampling. The partition is used to guide a sample warping algorithm to transform a uniform distribution of points so that they approximate the product distribution. The technique is unbiased, requires little precomputation, and we demonstrate that it works well for a variety of BRDF types. Further, we present an adaptive method which allocates varying numbers of samples to different image pixels to reduce shadow artifacts.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism

1. Introduction

Increasingly, image based lighting is being used for rendering. Image based lighting offers a number of advantages over simple lighting techniques such as directional or point lights. Spatially varying image based lighting provides a more realistic lighting environment, so images rendered with it often have a more realistic appearance. Using light probes as lighting allows virtual objects to be rendered as if they were actually located in the imaged location. This technique is employed often in movies, where rendered objects need to be seamlessly integrated into live action shots.

1.1. Importance Sampling for Direct Lighting

The problem that this paper addresses is how to solve the direct lighting equation. In a ray tracing context, to solve for direct lighting each primary ray must evaluate the integral:

$$L(x \rightarrow \Psi) = \int_{\Omega_x} L_e(x \leftarrow -\Theta) f_r(\Psi \leftrightarrow x \leftrightarrow \Theta) |\cos\theta| d\Theta. \quad (1)$$

In this equation, x is the point hit by the primary ray, L_e represents the incoming radiance at point x, f_r is the BRDF function, and θ is the angle between the surface normal at x and the outgoing direction, Θ.

Monte Carlo Integration. A common approach to evaluate the direct lighting equation is to use Monte Carlo integration, which replaces the continuous integral with the average of N Monte Carlo samples:

$$L(x \rightarrow \Psi) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(x \leftarrow -\Theta_j)\ f_r(\Psi \leftrightarrow x \leftrightarrow \Theta_j) |\cos\theta_j|}{p\ (\Theta_j)}. \quad (2)$$

Importance sampling attempts to minimize the variance of the above expression by choosing a sampling distribution to make the terms of the sum as constant as possible.

Light source and BRDF sampling. Many importance sampling techniques for direct lighting concentrate on either sampling the light source or the BRDF. For example, Burke [Bur04] described two methods to distribute samples according to the brightness of an environment map, based on cdf inversion and the alias method. Other research [KK03, ARBJ03, ODJ04] has tackled direct lighting from environment maps by approximating the illumination with a set of point lights (sampling directions). More recently, Debevec [Deb05] presented a simple technique to generate point lights to approximate environment lighting using a summed area table of the environment map. Our algorithm also uses a summed area table, but we are able to approximate the product distribution, not just the incident light term.

Methods that generate samples based solely on illumination do not work well for highly specular surfaces. In many situations it is more efficient to sample according to the BRDF function rather than the incident illu-

mination. Some analytical BRDFs can be directly importance sampled, including the Blinn model [Bli77], the Ward model [War92], the Lafortune model [LFTG97], and the Ashikhmin model [AS00]. Besides analytical BRDFs, importance sampling can be employed with sampled BRDFs. Lawrence, Rusinkiewicz and Ramamoorthi [LRR04] described a factored, tabular representation for BRDFs that is both compact and amenable to importance sampling.

Sampling a product distribution. Sampling according to one of the terms of the lighting equation can reduce variance, but it is more advantageous to generate samples based on all of the terms, rather than just one. Multiple importance sampling (MIS) [VG95] can sample the BRDF and lighting simultaneously, but the resulting distribution is more akin to the average of the terms rather than the product. Ideally, an importance sampling algorithm should be able to generate samples according to the product distribution.

Burke, Ghosh and Heidrich [BGH05] described a technique called bidirectional importance sampling (BIS) that can sample the product of an environment map and the BRDF, based on rejection sampling. The rejection sampling can be costly, however, and requires an unknown number of tries to produce samples from the product distribution. A second form of BIS that can produce samples in a deterministic amount of time replaces rejection sampling with resampling, but the resulting samples are only approximately distributed according to the product. Talbot, Cline and Egbert [TCE05] generalized this second form of BIS, placing it into the more general category of resampled importance sampling (RIS). Resampling methods can also be costly, however, since they rely on taking a large number of tentative samples, most of which will be discarded.

Recently, Clarberg et al. [CJAMJ05] presented an algorithm called Wavelet Importance Sampling (WaIS) that samples products of wavelet functions. Their algorithm uses a property of wavelets that allows a wavelet product to be evaluated in a top-down fashion, and they introduced a warping technique that transforms a uniform distribution of points to the product distribution using the wavelet product as a guide. WaIS produces very impressive results, but has a number of shortcomings that make it impractical in some situations. WaIS requires all BRDFs in the scene to be resampled as wavelets, which may be impractical for scenes with large numbers of BRDFs. Also, WaIS requires the wavelet functions to share a common coordinate system. In the case of environment maps, they accomplish this by storing a separate wavelet decomposition for each possible orientation of the environment map. Our two stage importance sampling algorithm can be thought of as a variant of WaIS that does not require a wavelet product to drive the sample warping. Instead, we use a hierarchical partitioning of the environment map similar to the probability trees described by McCool and Harwood [MH97].

1.2. Two Stage Importance Sampling

This section gives an overview of our new algorithm to perform importance sampling for direct lighting. The algorithm proceeds in two stages, so we call it *two stage importance sampling*, or just *two stage sampling*. The first stage creates an approximation to the product of the BRDF and the environment map suitable for importance sampling. The approximate product consists of (1) a summed area table of the environment map, times the cosine of the angle of inclination to compensate for foreshortening at the poles, along with (2) a hierarchical partition of the environment map annotated with BRDF values at the region corners. The second stage of the algorithm uses the summed area table and environment map partition to warp a set of uniformly distributed points so that they approximate the product distribution (BRDF $\times$ incident light). Both the hierarchical partitioning of the environment map and the sample warping are performed for each primary ray in a ray tracer. Figure 1 gives high level pseudocode for two stage importance sampling.

Two Stage Importance Sampling

Create a summed area table of the light source.
For each primary ray
1. Create an approximation to the product of the BRDF and the light source.
2. Warp a set of uniformly-distributed samples to to approximate the product distribution.

Figure 1: *Two stage importance sampling algorithm.*

Benefits of the new approach. Two stage importance sampling offers a number of benefits over existing techniques to sample according to the product distribution, such as RIS and WaIS. It requires very little precomputation, just a summed area table of the environment map. Furthermore, two stage sampling can work with both sampled and analytical BRDFs, and since BRDFs do not need any preprocessing, it can handle scenes with many BRDFs. Two stage sampling does not even require the ability to importance sample the BRDF function. All that is needed are the BRDF peaks. Since two stage sampling does not use rejection sampling or resampling, it preserves the stratification of an input sampling pattern better than algorithms that discard some of their samples, such as RIS. Finally, two stage sampling does not require the terms in the product to have the same coordinate system, so our algorithm could easily be adapted for other large light sources besides spherical environment maps.

2. Partitioning the Light Source

This section describes how to create a hierarchical partitioning of an environment map based on the BRDF function.

2.1. Environment Map Encoding

As a preprocessing step, we create a copy of the environment map as a summed area table [Cro84]. This allows rectangu-

lar regions within the map to be summed with just four table look-ups. Creating the summed area table typically requires less than 1 second. In our implementation, the summed area table stores the luminance of all of the static terms in the product (terms that are constant within the coordinate system of the environment map). For a spherical environment map in latitude, longitude format, we store the value in the environment map times the cosine of the angle of inclination, which compensates for the foreshortening that occurs near the map poles. This is the same encoding suggested by Debevec [Deb05]. To maintain precision, the summed area table is stored as 64 bit floating point values.

2.2. Partitioning the Environment Map

Two stage importance sampling relies on a piece-wise linear approximation of the BRDF and all of the terms of the product that are not included in the summed area table. This approximation is created independently for each primary ray. In the case of a spherical environment map, the terms not included in the summed area table are the BRDF and cosine from equation 1: $f_r(\Psi \leftrightarrow x \leftrightarrow \Theta)\,|\cos\theta|$. For brevity, we will refer to this value simply as f.

Our algorithm approximates the product function by partitioning the environment map into disjoint regions using an axis-aligned BSP tree. Nodes in the BSP hierarchy represent rectangular regions in the environment map, and each node stores the value of f at its four corners. Consequently, the approximation to the product becomes the product of the values stored in the summed area table and the bilinear interpolation of f from the leaf nodes of the BSP tree. Figure 2 shows one of these leaf nodes. Note that although the values of f are interpolated across the region, the values stored in the summed area table are represented exactly.

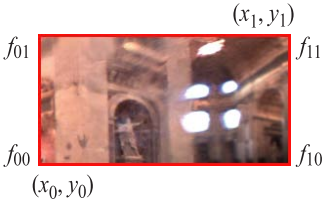

Figure 2: *Example region R, labeled with min and max extents along with f values at the corners.*

Creating the root node. The first step in partitioning the light source is to create a root node that will encompass all non-zero parts of the product function. One obvious choice would be to include the entire environment map as the root. However, we can easily find a smaller rectangle that encompasses all parts of the environment map that are in the same hemisphere as the surface normal. Assuming that the pixel coordinates of the normal in the environment map are (x_n, y_n), and w and h are the width and height of the environment map in pixels, the bounds of this rectangle are $(0,\ max(0,\ y_n - h/2))$ and $(w,\ min(h,\ y_n + h/2))$.

After creating the root node, we make an initial partition by subdividing at two locations. First, we subdivide the root node at the normal location, (x_n, y_n), and half the width of the environment map away from the normal, $((x_n + w/2)\%w,\ y_n)$. Subdividing at $((x_n + w/2)\%w,\ y_n)$ as well as the normal has the benefit that the $cos\theta$ term of the product is monotonic in all of the resulting regions. Since the hierarchy is a binary tree, we split at a particular location by first horizontally splitting the leaf node containing the position, and then vertically splitting both of the new nodes created by the first split. Figure 3 shows the initial partition.

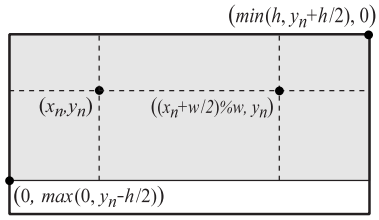

Figure 3: *Initial partition of the environment map. Two initial splits divide the root node into six regions, and include the peak of the* $\cos\theta$ *term in f as one of the region corners.*

Subdividing at the BRDF peaks. After the initial partition of the environment map has been made, we subdivide the leaf nodes at the peaks of the BRDF. The peak locations that we use for different BRDF types are as follows:

- **The Lambertian model** does not need any other peaks besides the normal direction.
- **The Oren-Nayar model** [ON94] was designed to simulate retro-reflection, so we add the incident direction as a peak for this model.
- **The Phong model** [Pho75] and **Blinn microfacet model** [Bli77] have a glossy lobe centered around the reflection vector, so we add the reflection vector as a peak for these models.
- **The Lafortune model** [LFTG97] consists of a number of generalized cosine lobes, each of which has a well-defined peak location. The peaks of the lobes are found by transforming the incident direction to the local shading coordinate system. The transformed incident direction can then be directly scaled based on the lobe parameters to produce the lobe peaks.
- **The Ashikhmin anisotropic model** [AS00] produces a long, thin specular lobe that traces out a cone of directions, rather than being centered around a single point. For this BRDF, we first add the reflection vector as a peak, since this is the brightest point on the BRDF. In addition, we add nine peaks along the cone of directions traced out by the specular lobe, if the anisotropy is large enough. In our implementation, we add the extra nine peaks of if n_u/n_v or n_v/n_u is greater than 3, where n_u and n_v are the anisotropy parameters of the BRDF. Figure 4 shows the local surface geometry for the Ashikhmin model where $n_u < n_v$. In the case where $n_u < n_v$, the specular lobe of the

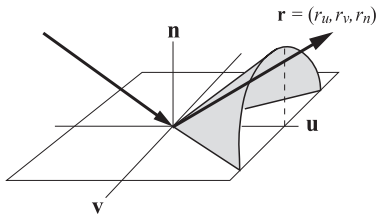

Figure 4: *The specular lobe of the Ashikhmin model stretches over a cone of directions rather than being centered around a single peak direction.*

BRDF stretches out into a cone of directions aligned with the **u** axis of the local coordinate system, which includes the reflection vector as the brightest point. We define nine "peak directions" to represent this conical lobe as follows: As in figure 4, let $(\mathbf{u},\mathbf{v},\mathbf{n})$ be the local coordinate system of the surface, and let (r_u, r_v, r_n) be the reflection vector defined in the local coordinate system. If $n_u < n_v$, the nine peaks, $peak_0 \ldots peak_8$, are defined by

$$peak_i = \left(r_u,\ cos\frac{\pi i}{8}\sqrt{r_v^2+r_n^2},\ sin\frac{\pi i}{8}\sqrt{r_v^2+r_n^2} \right),$$

If $n_v < n_u$, the conical specular lobe is aligned with the **v** axis rather than the **u**, and the nine peaks are defined by

$$peak_i = \left(cos\frac{\pi i}{8}\sqrt{r_u^2+r_n^2},\ r_v,\ sin\frac{\pi i}{8}\sqrt{r_u^2+r_n^2} \right).$$

Peaks for more general reflection models. For multi-component reflection models, we use all of the peaks defined by the components. For unknown or measured BRDFs, we start with the reflection and retro-reflection vectors and then importance sample the BRDF to find additional "peaks". In our implementation, we sample the BRDF $N/4$ times, where N is the number of samples to be taken. Each of the BRDF samples is then added as a new peak if the value of f at the sample location is greater than the average of f at the corners of the leaf node that the sample falls in.

Splitting region neighbors. Whenever a region is split, we check the neighboring regions that border the split location to see if they should be split as well. In particular, if the two corners of the neighbor region that are adjacent to the split have f values which sum to less than f_{split}, then

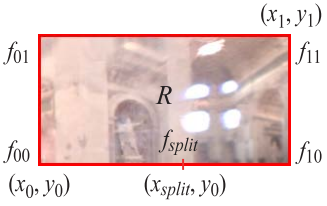

Figure 5: *Cascading region splits. In the diagram, the region below region R has just been split horizontally at location x_{split}. The region splitting algorithm cascades this split into region R if $f_{split} > f_{00} + f_{10}$.*

we cascade the split into the neighbor region. For example, suppose that the region directly below region R in Figure 5 was split horizontally at location x_{split}, with $x_0 < x_{split} < x_1$. Then if the value of f at (x_{split}, y_0) is greater than $f_{00} + f_{10}$, we split region R horizontally at location x_{split} as well. Note that in a spherical environment map the x axis wraps around, so that regions on the leftmost part of the environment map are neighbors of regions on the rightmost part. This extra splitting helps in the case of specular BRDF lobes that are not axis-aligned.

Subdividing based on split potential. At this point, we have a hierarchy that includes the peaks of the BRDF as region corners, but it still may not be a good approximation of the product. We can improve the approximation by splitting some of the regions that do not fit the product well.

For a region R with corners (x_0, y_0) and (x_1, y_1), and f values $f_{00} \ldots f_{11}$ (refer to Figure 5), we define a heuristic called the "split potential" that determines which region should be split to improve the product approximation the most. We base our heuristic on the idea that large and bright regions, and regions in which f varies a lot should be split first. More formally, we define the split potential, $p_{split}(R)$, as

$$p_{split}(R) = \sigma_f(R)\ sum(R)\ area(R), \tag{3}$$

where $sum(R)$ is the sum of the environment map over region R, $area(R)$ is the area of the region, and $\sigma_f(R)$ is the standard deviation of the f values at the corners of R:

$$\sigma_f(R) = \tfrac{1}{2}\sqrt{(f_{00}-f_{ave})^2+(f_{10}-f_{ave})^2+(f_{01}-f_{ave})^2+(f_{11}-f_{ave})^2}.$$

Our strategy is to perform a fixed number of node splits after creating the initial hierarchy, the same as the number of samples that will be taken, always splitting the leaf node with the highest split potential. This has the effect that each additional sample improves the sampling distribution, so that the convergence rate increases as more samples are added. Once again, whenever a region is split, its neighbors are checked to see if they should be split as well.

Split direction. In addition to deciding which region to split, the algorithm must decide whether to split the region along the x or y axis. One possibility would be to always split the region along the longest dimension, but we have found it better to choose the split axis based on the values of f as well as the axis lengths. Referring to Figure 5, a region will be split in the x direction if

$$\begin{aligned}&\left((f_{10}-f_{00})^2+(f_{11}-f_{01})^2\right)(x_1-x_0) > \\ &\qquad \left((f_{01}-f_{00})^2+(f_{11}-f_{10})^2\right)(y_1-y_0).\end{aligned} \tag{4}$$

Otherwise, the region will be split in the y direction.

Calculating region weights. As part of the sample warping algorithm, each region in the hierarchy must have an approximation to its contribution to the total product integral, which we call the region weight. For a leaf region L the weight is given by

$$weight(L) = sum(L)\ (f_{00}+f_{10}+f_{01}+f_{11})/4. \tag{5}$$

For a non-leaf region R, the weight is obtained by summing the weights of its children. As a convenience for the warping algorithm, we store two other values in non-leaf nodes in the hierarchy – the probability of choosing child A, $prob(A,R)$, and the fraction of the area that is taken up by child A, $areaFraction(A,R)$:

$$prob(A,R) = weight(A)/weight(R) \quad (6)$$

$$areaFraction(A,R) = area(A)/area(R). \quad (7)$$

Figure 6 gives pseudocode for the partitioning algorithm and shows example partitions for different BRDF types.

Environment map partitioning algorithm

For each primary ray
  Create the root node (Fig. 3).
  Split the root at (x_n, y_n) and $((x_n + w/2)\%w,\ y_n)$.
  Split leaves at BRDF peaks (and neighbors as needed).
  For $i = 1$ to # of samples per primary ray
    Find leaf node, R, that maximizes p_{split} (eq. 3).
    Split R along the axis specified by equation 4.
    Split neighbors of R as needed (Fig. 5).
  Calculate sampling weights for all regions (eq. 5 6 7).

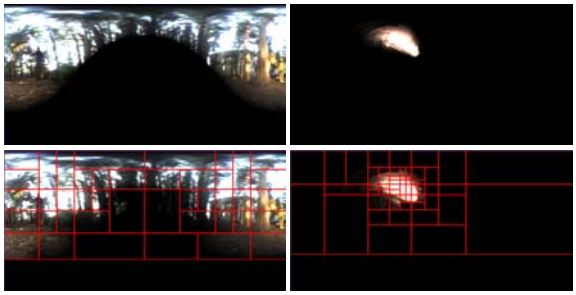

Figure 6: *Environment map partitioning algorithm and example partitions. The left images show the product of a highly diffuse Oren-Nayar BRDF in the Eucalyptus Grove environment (top), and the approximation made by the partitioning algorithm (bottom). The right images show a more specular BRDF that uses the Blinn microfacet model in Galileo's Tomb. In both cases, the partitioning algorithm produces a good approximation to the product. Note particularly how the split potential heuristic "homes in" on the specular lobe of the Blinn BRDF.*

3. Hierarchical Warping (revisited)

This section describes the warping algorithm that forms the second stage of two stage importance sampling. Our warping algorithm is based on the technique presented by Clarberg et al. [CJAMJ05] to transform a uniform distribution of points to the product distribution. The main difference between our algorithm and the one presented by Clarberg et al. is that in our algorithm warping is controlled by the environment map

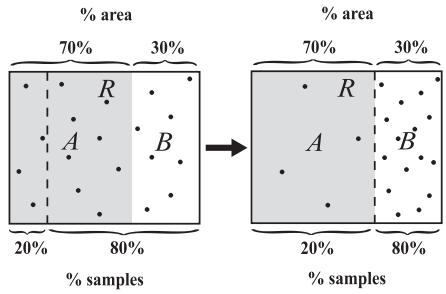

Figure 7: *One step of the sample warping algorithm. Here, an environment map region R has two children, with the left child, A, shown in gray, and the right child, B, shown in white. The samples allocated to R are originally distributed evenly (left image), but the sample warping stretches 20% of the samples to cover the child A, and the remaining 80% of the samples are squeezed to fit into child B (right image).*

partition described in section 2 rather than a wavelet product. Of course, this means that our algorithm only approximates the product distribution, but it also means that our algorithm requires minimal precomputation and storage (just a summed area table of the environment map). Also, since two stage importance sampling does not rely on a discrete sampling of the BRDF, it can work directly with analytical or sampled BRDFs in their native formats. Other differences include the fact that we warp one sample at a time rather than groups of samples, and the fact that regions do not always divide exactly in half in the environment map partition.

As an illustration of how the sample warping works, consider figure 7. The figure depicts a region, R, of the environment map that has two children. The left child of the region, A, shown in gray, contains 70% of the area of R, and the right child, B, shown in white, contains 30% of the area of R. Now, suppose that the algorithm wants to create a distribution that puts 20% of the samples in child A, and 80% of the samples in child B. The warping starts with a uniform distribution of samples, as shown in the left image. It then stretches the leftmost 20% of the samples to cover 70% of the area of R, and squeezes the remaining 80% of the samples to fit into 30% of the area, as shown in the right image. This process can then be repeated on A and B, and so on, to create any desired sampling distribution.

As previously stated, our warping algorithm guides the sample warping with the environment map partition described in section 2. The warping starts at the root node of the environment map partition with a sample (s,t) that is uniformly distributed in $[0,1)^2$ and a probability that is set to the area of the environment map divided by the area of the root node. The algorithm next decides which child of the root that the sample belongs to based on the value of $prob(A,R)$ stored in the root node. It then warps s or t depending on the split axis, and updates the probability for the sample. For simplicity, assume that the root is split on the x axis, so that

s should be warped. If $s \leq prob(A,R)$, then s is warped by dividing by $prob(A,R)$, transforming it into the local coordinate system of child region A. If $s > prob(A,R)$, s is set to $(s - prob(A,R))/(1 - prob(A,R))$, transforming it into the local coordinate system of child B. The sample probability is updated in a similar manner. If $s \leq prob(A,R)$, the sample probability is multiplied by $prob(A,R)/areaFraction(A,R)$, and if $s > prob(A,R)$, the sample probability is multiplied by $(1 - prob(A,R))/(1 - areaFraction(A,R))$, reflecting the change in probability density in the two child regions.

This warping process repeats on the chosen child node, and so on. Once the leaf level of the hierarchy is reached, a second warping routine takes over, warping the sample down to the pixel level in like manner, constructing virtual environment map regions below the leaf level to do the warping. Figures 13 and 14 in appendix A give code for the two sample warping routines. After a sample has been warped, it can be used directly to define a Monte Carlo sample for equation 2. Since the warping routine keeps track of the probability with which the sample direction was generated, Monte Carlo estimates made in this manner are unbiased.

4. Convergence to the Product Distribution

A note about our error metric. To test convergence we are using an error metric called the "coefficient of variation", which is defined as the standard deviation divided by the mean, σ/μ (RMSE / mean pixel value). Since this value is a scalar multiple of σ, it is just as valid a statistical measure as using σ or σ^2 directly. However, it also provides an intuitive metric for how much visual noise exists within an image. Based on our observations, a good rule of thumb seems to be that an image (with Gaussian noise spread evenly over the image plane) will be nearly flawless visually if σ/μ is less than about 0.01.

Pseudo-random number sequences. For all sampling methods in all of the experiments in the paper, we used Hammersley point sets that were randomly shifted to avoid correlation artifacts (called a Cranley-Patterson rotation [KK02, CP76]). The Hammersley point set has several properties that make it well suited for our application. First, it can create point sets of arbitrary size, not just powers of 2 or $n \times m$. Second, it consistently produced the lowest error of the sampling methods that we tried, including random and jittered point sets, and scrambled (0,2)-sequences.

Convergence of two stage sampling. To get a feel for how quickly two stage sampling converges to the product distribution, we rendered a number of spheres with different BRDFs using two stage importance sampling and multiple importance sampling in the Galileo's Tomb environment. Figure 8 shows the results of these experiments for three different BRDFs. Not only does our algorithm have a lower error than MIS for these examples, but the rate of convergence is much better as well. For example, two stage importance

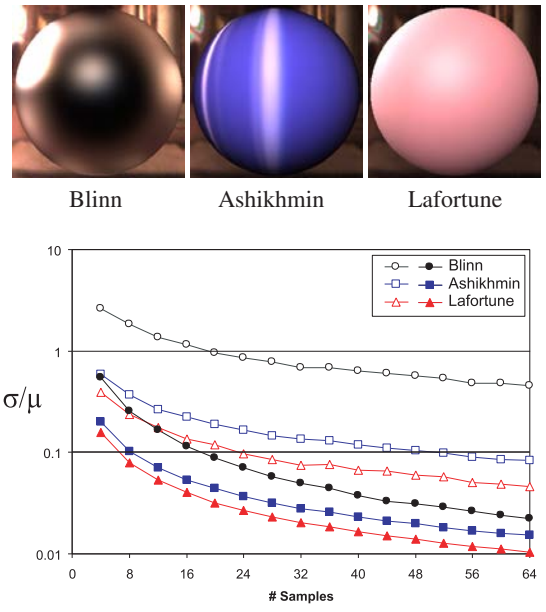

Figure 8: *Convergence of our algorithm (solid markers) vs. multiple importance sampling (hollow markers) for different BRDF types. The "Blinn" scene uses the Blinn microfacet model with a roughness of 0.02. "Ashikhmin" uses the Ashikhmin anisotropic model with u and v roughnesses of 0.001 and 1.0, respectively. Finally, "Lafortune" is a fit of the Lafortune model to measured skin reflectance.*

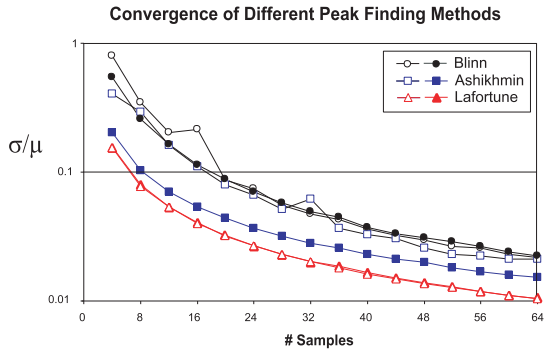

Figure 9: *Convergence of two stage sampling for different peak finding methods: sampling the BRDF distribution to find peak directions (hollow markers), and explicit enumeration of BRDF peaks as described in section 2.2 (solid markers).*

sampling converges at a rate of $N^{-1.17}$ for the Blinn scene, $N^{-0.93}$ for Ashikhmin, and $N^{-0.98}$ for Lafortune. Compare this to MIS, which converges according to $N^{-0.64}$, $N^{-0.70}$ and $N^{-0.77}$ for the same scenes, a much slower rate. Our algorithm is able to achieve such high convergence rates because increasing the number of samples actually improves the sampling distribution.

Convergence for different peak finding methods. Figure 8 demonstrates the convergence of two stage sampling when the BRDF peak directions are known. However, peak directions may not be easily computed for some reflection models, such as sampled BRDFs. In section 2.2, we advocated BRDF importance sampling as an alternative method to find peak directions when they are not easily computed otherwise. To test the effectiveness of this alternate procedure, we re-rendered the scenes from figure 8 using BRDF sampling to find the peak directions. To make the comparison fair, we did not include the reflection or retro-reflection vectors as initial peaks, since the Blinn and Ashikhmin model both have strong peaks around the reflection vector. Figure 9 plots the results of using BRDF sampling to find peaks (hollow markers) alongside the results from figure 8 for comparison. As can be seen in the figure, the Lafortune scene performs almost identically for both peak finding methods. The Blinn scene starts out a little worse with the alternate peak finding method, but quickly overtakes the performance of the standard peak finding method. The Ashikhmin scene, on the other hand, does not perform as well with the alternate peak finding method, likely because of the complexity of the conical specular lobe, but it still outperforms MIS by a wide margin. These results suggest that BRDF importance sampling can be an effective general peak finding method for many BRDF types.

5. Variable Samples

Section 4 describes the performance of two stage importance sampling in the absence of shadows. However, most scenes contain shadows and other features that pose difficulties for a Monte Carlo sampler. Penumbra regions are particularly prone to noise. A desirable goal would be to assign different numbers of samples to different parts of the image to distribute noise more or less evenly over the entire image plane.

We achieve this goal by allocating more samples to pixels that are likely to have a high variance, and less samples to pixels likely to have a low variance. To do this, we need an estimate of the standard deviation of samples for the current pixel, σ_p, and the average standard deviation over all pixels, σ_{ave}. We obtain σ_{ave} in a preprocessing step, computing the average standard deviation for a small random subset of the pixels in the image (1024 in our implementation). σ_p is computed as a running average over previous samples. It's value is set to zero initially, and after a pixel (primary ray) has been processed, the value is updated using the formula

$$\sigma_p^{new} = 0.75\sigma_p + 0.25\sigma_{pixel} \tag{8}$$

where σ_{pixel} is the sample variance of the pixel just processed. This process is similar to the variance tracking method used for efficiency optimized Russian roulette in bidirectional path tracing [VG94], except that we are using the value for a different purpose.

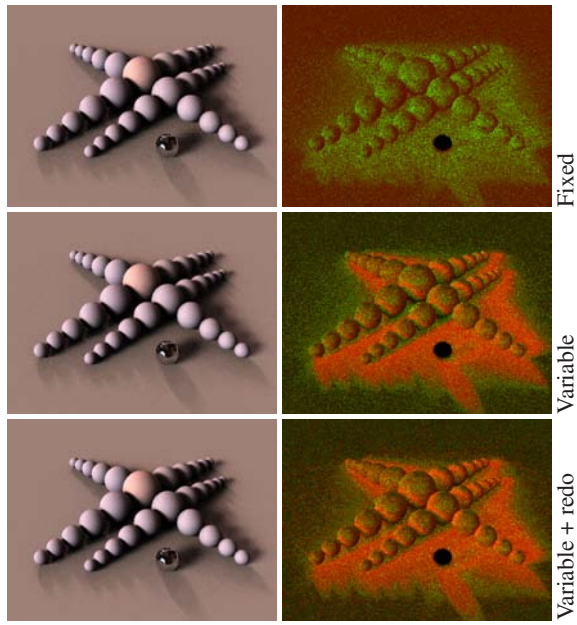

Figure 10: *Fixed and variable sampling rates. The left column of images shows renderings made with different sampling strategies in the Galileo's Tomb environment, and the right column shows error in green and the number of samples allocated to different pixels in red (see color plate). All renderings use about 16 samples per pixel on average. Fixed rate sampling concentrates errors in the penumbra regions of the image ($\sigma/\mu = 0.0446$, 25.5 sec.). Variable sampling spreads the error more evenly, and overall error is reduced ($\sigma/\mu = 0.0409$, 25.9 sec.). Note how samples concentrate in the high error regions. Finally, re-rendering the few pixels that significantly overshoot their variance estimates gets rid of most of the highest error pixels ($\sigma/\mu = 0.0366$, 27.3 sec.).*

At this point, we need to determine a number of samples to use for the next pixel. To make the sample allocation robust, the number of samples for a pixel is always set to at least half of what it would be in the fixed case. For the other half of the samples, the algorithm must decide how many to allocate to each pixel based on σ_p. We could use the variance analysis results from Mitchell [Mit96] or Kollig and Keller [Mit96] as a guide to determine how many samples to allocate to different pixels, but this would lead to a different expression for each input sequence type. Instead, we make the rather crude assumption that the standard deviation will decrease as N^{-1}, and allocate the second half of the samples accordingly. Thus, the number of samples that we allocate to a pixel is

$$M = \frac{N}{2}\left(1 + \frac{\sigma_p}{\sigma_{ave}}\right), \tag{9}$$

where N is the average number of samples that should be allocated to each pixel. Reallocating samples in this way tends

to decrease statistical error slightly, but perceptual error is reduced much more, since the sample reallocation spreads the error more or less evenly over the image plane. Allocating variable samples as just described does not introduce bias since information from the current pixel does not influence the variance estimate, σ_p [KA91].

A problem that can occur when using variable samples is that the variance estimate, σ_p, may not be accurate. This can happen at object edges, for instance. Our solution is to re-render those few pixels (about 5%) in which the actual variance within the pixel is much greater than the estimate. Specifically, if $\sigma_{pixel} > \sigma_p + \sigma_{ave}$, we reset σ_p to σ_{pixel}, and render the pixel again. Although this step adds a slight bias to the solution, it is quite adept at eliminating most of the highest error pixels. Figure 10 shows a scene rendered with fixed and variable numbers of samples.

6. Results

We have implemented two stage importance sampling as an extension to the PBRT [PH04] rendering system. PBRT supports a variety of analytical BRDF models which we used to validate our algorithm, including the Blinn, Ashikhmin, Lafortune and Oren-Nayar models.

Comparison with other robust sampling algorithms. Figure 12 compares our two stage sampling algorithm to multiple importance sampling (MIS), and resampled importance sampling (RIS). For RIS, we use multiple importance sampling to generate 2 tentative samples per accepted sample. The number of tentative samples per accepted sample in our study, 2, was chosen by trial and error to maximize image quality vs. render time. Burke, Ghosh and Heidrich [BGH05] were able to quickly generate dozens of tentative samples per accepted sample using the Phong model. We have found the expense of generating samples for arbitrary BRDFs and evaluating them without visibility in PBRT to be comparatively large, about $1/3$ the cost of shadow queries. This is more in line with the results reported by Talbot, Cline and Egbert [TCE05].

We do not presently have an implementation of wavelet importance sampling to directly compare with our algorithm, but we can simulate WaIS using our algorithm by greatly increasing the number of region splits in the environment map. The resulting probability distribution comes very close to the actual product distribution, and thus mimics WaIS quite well. WaIS render times were approximated simply by copying the timing results from MIS. Note that the timing results do not include the substantial startup costs of WaIS. As can be seen in the table of images, two stage importance sampling displays a lot of noise at very low sample counts, but it quickly converges. Although we cannot expect to surpass the convergence rate of WaIS, our algorithm does start to become competitive with it in terms of quality per number of samples after just a few dozen samples per pixel, without the long preprocessing times needed for WaIS.

Figure 11: *Scene with a spatially varying specular exponent. This scene literally contains hundreds of different BRDF shapes. The scene was rendered with 4 primary rays per pixel and 32 two stage samples per primary ray at a resolution of* 1024×512*. Render time was 895 seconds.*

Scenes with large numbers of BRDFs. Figure 11 shows a scene in which the specular roughness parameter is controlled by a texture map. This scene would be impractical for WaIS because each surface point potentially has a BRDF with a different shape. By contrast, two stage importance sampling has no difficulties with the scene because it does not need to preprocess BRDFs.

7. Conclusions and Future Work

In this paper we have presented a new importance sampling technique for direct lighting called *two stage importance sampling*. The technique is unbiased, and we showed that it works well for scenes with complex BRDFs and spatially varying environment map lighting. We showed that two stage importance sampling outperforms multiple importance sampling (MIS) and resampled importance sampling (RIS) in a number of rendering contexts, and that the new algorithm compares favorably with wavelet importance sampling in terms of quality per number of samples at fairly low sample densities, while requiring substantially less precomputation. In addition, we described a novel technique to allocate variable numbers of samples to different pixels in a rendered image to reduce noise in shadow areas, and showed that the new sample allocation technique can decrease statistical and visual error in rendered images.

There is a fairly large overhead associated with partitioning the light source, and this limits the number of primary rays that can be cast per pixel. It would be interesting to cast a larger number of primary rays, and then use RIS to limit the number of product approximations that must be evaluated. Another idea would be to reuse partitions between primary rays when the BRDF setup is similar enough. Currently, our product approximation does not account for color information because the f values and the summed area table only store luminances. We could incorporate color into our algorithm by storing a color summed area table and color f values. Our current implementation uses a very simple rule

to determine the number of regions to split when partitioning the environment map (the number of splits equals the number of samples). There may be better heuristics for the number of splits in terms of image quality vs. time. Finally, we note that while our algorithm to allocate different numbers of samples to different pixels is useful, it does not take directional information into account. As the algorithm now stands, shadow regions account for the majority of the noise at sample rates above a few dozen per pixel, so improved anti-aliasing for shadow regions would be of great benefit.

Acknowledgements. We would like to thank Kate Kuttler and T-Splines, LLC, for the iguana model, Paul Debevec for the environment maps used in the paper, and as always, the Stanford scanning repository for the the bunny model.

References

[ARBJ03] AGARWAL S., RAMAMOORTHI R., BELONGIE S., JENSEN H. W.: Structured importance sampling of environment maps. *ACM Trans. Graph. 22*, 3 (2003), 605–612.

[AS00] ASHIKHMIN M., SHIRLEY P.: An anisotropic phong BRDF model. *J. Graph. Tools 5*, 2 (2000), 25–32.

[BGH05] BURKE D., GHOSH A., HEIDRICH W.: Bidirectional importance sampling for direct illumination. In *Rendering Techniques 2005 Eurographics Symposium on Rendering* (Aire-la-Ville, Switzerland, 2005), Bala K., Dutré P., (Eds.), Eurographics Association, pp. 139–146.

[Bli77] BLINN J. F.: Models of light reflection for computer synthesized pictures. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (NY, USA, 1977), ACM Press, pp. 192–198.

[Bur04] BURKE D.: Bidirectional importance sampling for illumination from environment maps. *Master's Thesis, University of British Columbia* (2004).

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Trans. Graph. 24*, 3 (2005), 1166–1175.

[CP76] CRANLEY R., PATTERSON T. N. L.: Randomization of number theoretic methods for multiple integration. *SIAM Journal of Numerical Analysis 13* (1976), 904–914.

[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (NY, USA, 1984), ACM Press, pp. 207–212.

[Deb05] DEBEVEC P.: A median cut algorithm for light probe sampling. *SIGGRAPH 2005 Poster* (2005).

[KA91] KIRK D., ARVO J.: Unbiased sampling techniques for image synthesis. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (NY, USA, 1991), ACM Press, pp. 153–156.

[KK02] KOLLIG T., KELLER A.: Efficient multidimensional sampling. *Computer Graphics Forum 21*, 3 (Sept. 2002), 557–563.

[KK03] KOLLIG T., KELLER A.: Efficient illumination by high dynamic range images. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 45–50.

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 117–126.

[LRR04] LAWRENCE J., RUSINKIEWICZ S., RAMAMOORTHI R.: Efficient BRDF importance sampling using a factored representation. *ACM Trans. Graph. 23*, 3 (2004), 496–505.

[MH97] MCCOOL M. D., HARWOOD P. K.: Probability trees. In *Graphics Interface '97* (1997), Davis W. A., Mantei M.,, Klassen R. V., (Eds.), Canadian Human-Computer Communications Society, pp. 37–46.

[Mit96] MITCHELL D. P.: Consequences of stratified sampling in graphics. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (NY, USA, 1996), ACM Press, pp. 277–280.

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.: Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph. 23*, 3 (2004), 488–495.

[ON94] OREN M., NAYAR S. K.: Generalization of lambert's reflectance model. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (NY, USA, 1994), ACM Press, pp. 239–246.

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

[Pho75] PHONG B. T.: Illumination for computer-generated pictures. *Communications of the ACM 18*, 6 (1975), 311–317.

[TCE05] TALBOT J. F., CLINE D., EGBERT P. K.: Importance resampling for global illumination. In *Rendering Techniques 2005 Eurographics Symposium on Rendering* (Aire-la-Ville, Switzerland, 2005), Bala K., Dutré P., (Eds.), Eurographics Association, pp. 139–146.

[VG94] VEACH E., GUIBAS L. J.: Bidirectional Estimators for Light Transport. In *Rendering Techniques 1994 (Proceedings of the Fifth Eurographis Workshop on rendering)* (1994), pp. 147–162.

[VG95] VEACH E., GUIBAS L. J.: Optimally combining

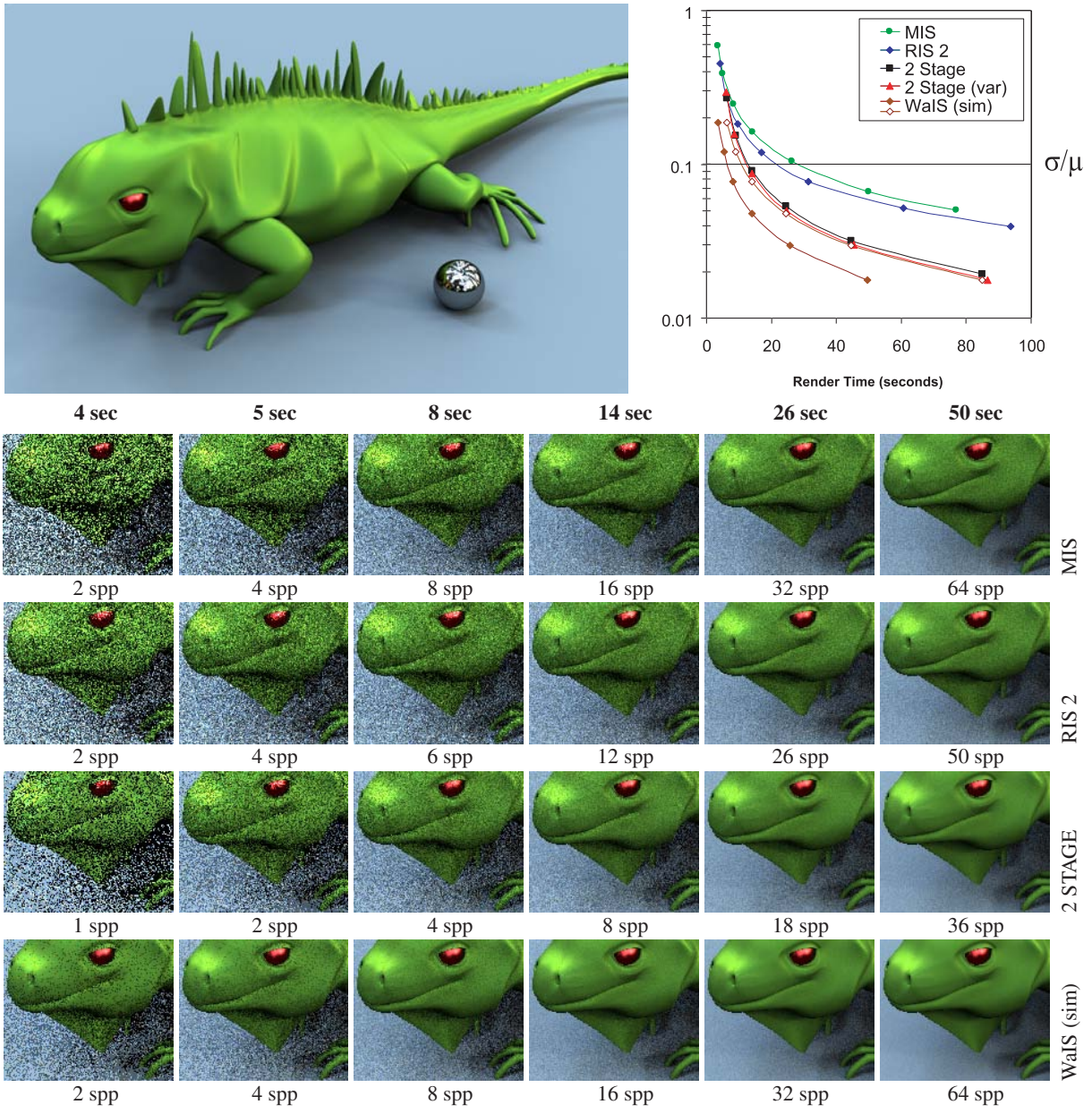

Figure 12: *Convergence of two stage importance sampling in terms of render time. The chart plots image quality against render time for multiple importance sampling (MIS), resampled importance sampling with two tentative samples per accepted sample (RIS 2), our two stage importance sampling algorithm (2 Stage), and a simulated version of wavelet importance sampling (WaIS). For comparison, we have plotted WaIS against estimated render time (solid diamonds), and using the same timing as two stage sampling (hollow diamonds). Although WaIS beats our algorithm in terms of quality per unit time (if we don't count startup time) this is mainly because WaIS can generate samples more quickly. When we compare using number of samples instead of render time (hollow diamonds), our algorithm starts out worse, but quickly converges to within a few percent of the noise level of WaIS. Note also that render times for WaIS do not include preprocessing. A major advantage of our algorithm over WaIS is that we have extremely low startup time (less than a second), whereas Clarberg et al. [CJAMJ05] reported startup times of more than an hour for some scenes. The table of images gives a quality comparison for approximately equal render times.*

sampling techniques for monte carlo rendering. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (NY, USA, 1995), ACM Press, pp. 419–428.

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques* (NY, USA, 1992), ACM Press, pp. 265–272.

Appendix A: Sample warping routines.

```
void warpSample1(float &s, float &t, float &prob, Region *R,
        SummedAreaTable &sumTable)
{
    prob = sumTable.width * sumTable.height / (float)R->area();
    while (R->hasChildren()) {
        if (R->splitAxis == X_AXIS) {   // SPLIT ON X AXIS
            if (s < R->probA) {   // S IS ON MIN X SIDE
                s /= R->probA;
                prob *= R->probA / R->areaFractionA;
                region = R->childA;
            } else {   // S IS ON MAX X SIDE
                s = (s − R->probA) / (1 − R->probA);
                prob *= (1 − R->probA) / (1 − R->areaFractionA);
                R = R->childB;
            }
        } else {   // SPLIT ON Y AXIS
            .
            .  // Y CASE OMITTED FOR BREVITY
            .
        }
    }
    warpSample2(s, t, prob, R->x0, R->y0, R->x1, R->y1,
            R->f00, R->f10, R->f01, R->f11, sumTable);
}
```

Figure 13: *Warping Algorithm (part 1). The WarpSample1 routine is called with s and t uniformly distributed in* $[0,1)^2$*, and R set to the root node in the hierarchy. When warpSample1 returns, s and t are still in* $[0,1)^2$*, but are now approximately distributed according to the product distribution, and prob contains the probability that the resulting point was chosen in terms of solid angle.*

```
void warpSample2(float &s, float &t, float &prob,
        int x0, int y0, int x1, int y1,
        float f00, float f10, float f10, float f11,
        SummedAreaTable &sumTable)
{
    float f0mid, f1mid, fmid0, fmid1;
    float faveA, faveB, areaFractionA;
    float sumA, sumB, probA, cosPhi;
    int xmid, ymid;

    while ((x1−x0) > 1 || (y1−y0) > 1) {
        if (x1−x0 > y1−y0) {   // SPLIT ON X AXIS
            .
            .  // X CASE OMITTED FOR BREVITY
            .
        } else {   // SPLIT ON Y AXIS
            ymid = (y0+y1)/2;
            areaFractionA = (ymid−y0) / (float)(y1−y0);
            f0mid = f00 + areaFractionA * (f01−f00);
            f1mid = f10 + areaFractionA * (f11−f10);
            faveA = f00 + f10 + f0mid + f1mid;
            faveB = f0mid + f1mid + f01 + f11;
            sumA = sumTable.sum(x0, y0, x1, ymid) * faveA;
            sumB = sumTable.sum(x0, ymid, x1, y1) * faveB;
            probA = sumA / (sumA+sumB);
            if (t < pA) {   // T IS ON MIN Y SIDE
                t = t / probA;
                prob *= probA / areaFractionA;
                y1 = ymid;
                f01 = f0mid;
                f11 = f1mid;
            } else {   // T IS ON MAX Y SIDE
                t = (t − probA) / (1 − probA);
                prob *= (1 − probA) / (1 − areaFractionA);
                y0 = ymid;
                f00 = f0mid;
                f10 = f1mid;
            }
        }
    }
    // CONVERT S AND T TO GLOBAL COORDINATES
    s = (s+x0) / sumTable.width;
    t = (t+y0) / sumTable.height;
    // CONVERT TO PROBABILITY OVER SOLID ANGLE
    cosPhi = cos( PI * (0.5 − t) );
    prob *= cosPhi / (2.0 * PI * PI);
}
```

Figure 14: *Warping Algorithm (part 2). The warpSample2 warps* (s,t) *down to the pixel level. After the warping is done, s and t are converted to the global coordinate system of the environment map. prob is also converted from probability over the area of the environment map to be in terms of solid angle, multiplying by* $\cos\phi/2\pi^2$*, where* ϕ *is the angle of inclination to point* (s,t) *in the environment map.*