Importance Sampling of Reflections from Hair Fibers

Pixar Technical Memo 11-07

Christophe Hery[1] and Ravi Ramamoorthi[2]

[1]Pixar Animation Studios [2]University of California, Berkeley

Abstract. Hair and fur are increasingly important visual features in production rendering, and physically-based light scattering models are now commonly used. In this paper, we enable efficient Monte Carlo rendering of reflections from hair fibers by describing a simple and practical importance sampling strategy for the reflection term in the Marschner hair model. Our implementation enforces approximate energy conservation, including at grazing angles by modifying the samples appropriately, and includes a Box-Muller transform to effectively sample a Gaussian lobe. These ideas are simple to implement, but have not been commonly reported in standard references. Moreover, we have found them to have broader applicability in sampling surface specular BRDFs.

1. Introduction

Hair and fur are important visual features, that are increasingly common in production environments. They are also the building blocks for accurate rendering of seemingly unrelated effects such as clothing, where we model individual fibers of yarn. Standard surface reflection algorithms no longer apply directly, since a hair fiber does not have a surface normal in the conventional sense, but only an orientation or tangent direction.

For many years, the standard hair reflection model was the extension of the Phong model proposed by [Kajiya and Kay 89]. This model was adapted for production by [Goldman 97]. In 2003, [Marschner et al. 03] proposed a comprehensive physically-based light scattering model from human hair fibers, that has become the basis for most subsequent work, including this paper. While the Marschner model defines an effective hair "BRDF", efficient Monte

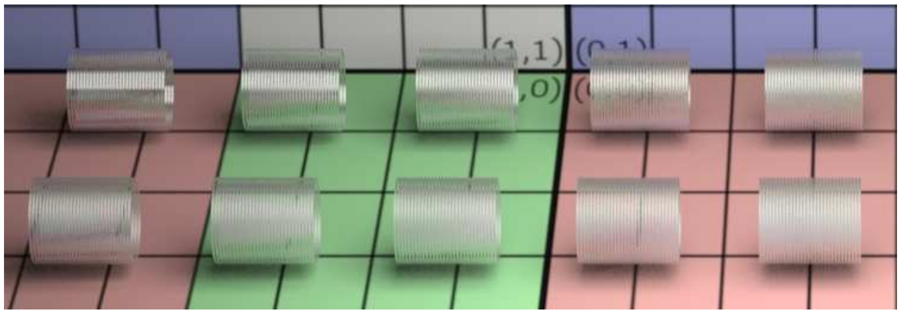


Figure 1: Our importance sampling method for the Marschner specular lobe BRDF is used inside a global illumination renderer for both direct lighting from an area light source and for tracing reflections. These images use 48 samples. Each cylinder is composed of 30 hair fibers shaped like circles and assembled next to one another. From left to right, top to bottom, the cone angle β is 1, 3, 5, 7, 9, 11, 13, 15, 17 and 19°.

Carlo rendering also requires practical techniques for importance sampling (we use BRDF importance sampling [Lawrence et al. 04] within a multiple importance sampling framework [Veach and Guibas 95]). To date, no importance sampling method has been published, and personal communications indicate the lack of widespread existence of such a method.

In this paper, we describe a simple and practical importance sampling scheme for the single scattering or reflection term R in the Marschner hair model. While we do not address the other (TT and TRT) terms, they are often considered separately for easier artistic design [Sadeghi et al. 10], and have a similar form. Importance sampling is by now a well-researched technique, and most of the key ideas are already in the literature. The contribution of this paper is to present practical solutions to a number of important details.

Our method was originally developed simultaneously with the recent comprehensive work by [d'Eon et al. 11] and addresses some of the same issues. In particular, they address issues of energy conservation, modifying the BRDF accordingly. However, the resulting model is fundamentally different in form from the gaussian lobes of the original Marschner paper and may not correspond to the physical characteristics of light reflection based on microfacets. Most significantly, importance sampling is still not addressed.

In summary, we describe a simple practical approach to importance sampling the reflected lobe in the Marschner hair model. We enforce approximate energy conservation at grazing angles by clamping and simplifying the weight of the Monte Carlo estimator, which is a "trick" that is also useful in other contexts like specular BRDF sampling. There are a number of interesting practical issues, that we describe in detail with pseudocode faithful to our actual production-ready implementation.

The panel in Figure 1 shows the results of our methods: with a fairly low number of samples, each hair fiber is reflection tracing to its surrounding walls and ground. Observe that we can resolve both glossy and semi-glossy specular defined by the Marschner model.

2. Background

The reflected radiance is given in the standard way by

$$L_r(\omega_r) = \int L_i(\omega) S(\omega, \omega_r) V(\omega) \cos\theta \, d\omega, \tag{1}$$

where the integral is over all incident directions, S is the scattering function (equivalent to the BRDF) for hair, L_i is the lighting, and ω and ω_r are incident and reflected directions. V is the visibility function that is ray traced, or computed using an approach like deep and multilayer shadow maps [Lokovic and Veach 00, Xie et al. 07]. For notational simplicity, incident directions are not subscripted. The important difference from surface reflection is that the angles are measured with respect to the normal *plane* (perpendicular to the hair tangent direction), rather than a single surface normal. Thus, θ is the incident angle to the plane, which ranges from $[-\pi/2, \pi/2]$.

Our goal is to importance sample $S(\omega, \omega_r)$ to determine incident directions, given that we know the reflected direction ω_r. In doing so, we will pick a number J of samples ω_j, and compute

$$L_r \approx \frac{1}{J} \sum_{j=1}^{J} \frac{L_i(\omega_j) S(\omega_j, \omega_r) V(\omega_j)}{p(\omega_j)} \frac{\cos\theta_j}{\cos^2\theta_{d,j}}, \tag{2}$$

where in the standard way, the Monte Carlo estimator takes the value of the integrand divided by the probability $p(\omega_j)$ of generating the sample. For reasons of notational simplicity in later derivations, we include the $\cos^2\theta_d$ term in [Marschner et al. 03] in the denominator above, outside of S. We will see that this term approximately cancels, and in any event we do not attempt to importance sample it.

To do the importance sampling, we need to know the form of the scattering function, which is given in [Marschner et al. 03] by

$$S(\theta_i, \phi_i, \theta_r, \phi_r) = M(\theta_i, \theta_r) N(\phi_i, \phi_r). \tag{3}$$

This form is already factored, allowing us to use many of the techniques for BRDF importance sampling in [Lawrence et al. 04]. Note that we have not explicitly considered the Fresnel term, nor the division by $\cos^2\theta_d$ (included directly in equation 2). If the Fresnel term is desired, it can simply multiply the value of the estimator (numerator in equation 2), but we do not consider it in the importance sampling itself.

Finally, for the reflection lobe, we use the formulae,

$$\begin{aligned} M &= g(\beta, \theta_h - \alpha) \\ N &= \cos\frac{\phi}{2}, \end{aligned} \tag{4}$$

where g is a (normalized) gaussian lobe, θ_h is the half-angle between incident and reflected directions, α is an offset to capture the shift in reflected angle because of the tilt of the surface scales, and $\phi = \phi_r - \phi_i$ is the azimuthal angle in the range $[-\pi, \pi]$. The form for M is directly from [Marschner et al. 03], while the form for N is a common simplification that can be derived [Sadeghi et al. 10]. Note also that unlike surface reflection, we are considering angles to the normal plane, so the formula for θ_h is just $\theta_h = (\theta_i + \theta_r)/2$.

This paper now describes how to generate sample directions in a probability distribution corresponding to the scattering function in equation 3, and how to compute equation 2.

3. Sampling

This section is the main body of the paper and describes how to generate the ω_j samples, and assign their directions, probabilities and weights of the Monte Carlo estimator in equation 2. Section 4 discusses some refinements needed for multiple importance sampling. We include pseudocode for the entire process, in Algorithm 1. Our system is implemented as a RenderMan shader, and the pseudocode is taken directly from our source code, with only minor editing for readability and to conform to the notational conventions in the text.

3.1. Basic Setup

The basic sampling function definition takes as inputs the reflected direction ω_r, an array of random numbers Rnd (each element is a vector since as we shall see, we will need 3 independent random numbers), and a structure for the geometry (that will be used to transform into local coordinates later). The output will be the BRDF samples (their directions, weights and probabilities for computing the estimator). The random numbers can be generated in the standard way, with stratified or quasi-Monte Carlo methods. ω_r is assumed to be available in a local coordinate frame aligned with $u - v - w$ directions as in [Marschner et al. 03], where u is the tangent along the hair, and v and w represent the normal plane. We first compute θ_r and ϕ_r.

Algorithm 1. (Algorithm Pseudocode for Generating BRDF samples)

```
public void sample (vector $\omega_r$; vector Rnd [ ]; in Geom; out BRDFsamp)
// Basic Setup, compute $\theta_r$ and $\phi_r$
float $\theta_r = \frac{\pi}{2} - \mathtt{acos}(\omega_r[0])$ ;
float $\phi_r = \mathtt{atan}(\omega_r[2], \omega_r[1])$ ;
// Now, loop over the required number of samples
uniform float k ;
$\theta_{\max} = \pi/2 - \mathtt{abs}(\theta_r/2 - \alpha)$ ;
for $(k = 0 ; k <$ numDirections; $k$ += $1)$ do
    vector Rand = Rnd[k] ;
    // Box-Muller Transform for sampling $M$
    float $\theta_s = \beta$ * sqrt (-2.0 * log (Rand[0])) * cos ($2\pi$ * Rand[1]) ;
    // Account for edge conditions
    if (abs $(\theta_s) > \theta_{\max}$) then
        $\theta_s$ = sign $(\theta_s)$ * $\theta_{\max}$ ;
    end
    $\theta_h = \theta_s + \alpha$ ; // Account for tilt from cuticle scales
    $\theta_i = 2.0$ * $\theta_h$ - $\theta_r$ ; // Convert to $\theta_i$
    if (abs $(\theta_i) > \pi/2$) then
        $\theta_i$ = sign $(\theta_i)$ * ($\pi$ - abs $(\theta_i)$) ; // Set $\theta_i$ to $[-\pi/2, \pi/2]$
    end
    float cosi = cos $(\theta_i)$ ; // Frequently used trig function
    // Inverse-CDF for $N$ and generate sample direction
    float $\triangle\phi$ = 2.0 * asin (2.0 * Rand[2] - 1.0) ;
    float $\phi_i = \phi_r + \triangle\phi$ ;
    vector $\omega_i$ = vector (sin $(\theta_i)$, cosi * cos $(\phi_i)$, cosi * sin $(\phi_i)$) ;
    BRDFsamp $\rightarrow$ dir[k] = Geom $\rightarrow$ transformFromLocal($\omega_i$) ;
    // Sample weights and pdf
    uniform float denom = -0.5 / $\beta$ / $\beta$ ;
    float $M = \frac{1}{\beta\sqrt{2\pi}}$ * exp ($\theta_s$ * $\theta_s$ * denom) ;
    float $N$ = 2.0 * sqrt (Rand[2] * (1.0 - Rand[2])) ; // cos asin (u)
    BRDFsamp $\rightarrow$ pdf[k] = $M$ * $N$ / (8.0 * cosi) ;
    // If desire $\cos\theta_d$:  cosd = $\max(\cos((\theta_i - \theta_r)/2), 1.0e-5)$
    // BRDFsamp $\rightarrow$ wt[k] = $K_s$*(cosi * cosi) / (cosd * cosd);
    // Simpler practical form, that conserves energy below
    BRDFsamp $\rightarrow$ wt[k] = $K_s$ ;
end
```

3.2. Box-Muller for Sampling the Gaussian for M

We begin by generating samples according to the gaussian for the M term. The standard approach is based on an inverse-cumulative distribution function. However, the gaussian cannot be analytically integrated and inverted, which means we would need to resort to computing the *inverse erf* function or numerical inversion. While *erf* is a standard numerical function in most packages including RenderMan, the *inverse erf* is found in Mathematica and Matlab, but is not standard in most shading languages, including RenderMan. There are a number of routines to compute the *inverse erf* [Acklam 03], but they can be expensive and difficult to port. Instead, there is a simple trick using two random variables known as the Box-Muller transform [Box and Muller 58]. Consider first a standard 2D normal distribution,

$$\begin{aligned} \text{pdf} &= \tfrac{1}{2\pi}\exp\left[-\tfrac{(x^2+y^2)}{2}\right] &&= \frac{1}{2\pi}\exp\left[-\frac{R^2}{2}\right] \\ \text{cdf} & &&= \exp\left[-\frac{R^2}{2}\right]. \end{aligned} \tag{5}$$

where R is the radius in polar coordinates, corresponding to cartesian coordinates of (x, y) and we consider the canonical case with variance 1. Polar coordinates are easier to work with in integrating to find the CDF, since the differential area measure is $R\,dR\,d\Theta$. The standard inverse-CDF method now allows us to sample, given uniform random variables U_1 and U_2 in $[0\ldots 1]$,

$$\begin{aligned} R^2 &= -2\log U_1 & \Theta &= 2\pi U_2 \\ X &= \sqrt{-2\log U_1}\cos 2\pi U_2 & Y &= \sqrt{-2\log U_1}\sin 2\pi U_2. \end{aligned} \tag{6}$$

While the above derivation is for the 2D normal distribution, each random variable X or Y is also a normal distribution, and we can use either for the 1D gaussian for M. Note that unlike standard inverse-CDF methods, we are using two random numbers to generate a single sample. There are also rejection sampling-based methods to avoid the trigonometric calculations, but we did not use them in our implementation. The wikipedia page on the Box-Muller transform [Box and Muller 58] has an excellent discussion of the alternatives.

In line 8 of the algorithm, we first sample the 1D gaussian to generate θ_s. Note that the basic Box-Muller value is multiplied by β to account for the variance. To obtain θ_h, we will now account for the offset α and edge effects.

3.3. Accounting for Edge Cases

The normal distribution function or gaussian has no limits on its range, but angles must generally be within the $[-\pi, \pi]$ range. This leaves the question of how to handle samples that lead to angles outside these limits (not so much a problem for θ_s itself, but for the result in θ_i). Note this only occurs in the tail of the gaussian and so any suitable method will generally lead to only minimal bias. However, these edge cases must be addressed explicitly in some way to avoid generating numerical garbage.

One physically-based approach [Lawrence et al. 04] is to simply set the weight for these samples to 0. While this is physically accurate for the BRDF as written, it leads to some potentially undesirable properties with losing some incident energy; a contant white dome with a specular albedo of 1 should ideally reflect an energy of 1, but setting samples to 0 loses energy. An alternative would be to reject those samples and renormalize the weight of the remaining samples. This approach is reasonable, but wastes samples.

Therefore, in practice, we impose a maximum value on the gaussian and clamp to that range. This only affects samples deep in the tail, and this clamping introduces minimal bias (we discuss conditions under with the approximation is reasonable below). More sophisticated changes to the gaussian function itself, to handle this in a more principled fashion, as in [d'Eon et al. 11] are a subject of future work.

In particular, in line 5, we compute the maximum value for θ_s to ensure that $\mid \theta_i \mid < \pi$ (simple algebra will verify the result), and in line 10 we clamp the sampled value to this maximum. Only then do we compute the values for θ_h and finally θ_i.

We still have more work to do in dealing with edge cases, since θ_i is a valid angle and therefore geometric direction, but may not lie in the $[-\pi/2, \pi/2]$ range that is required by our conventions. The pseudocode in line 15 shows one of many possible ways of handling this. For the paranoid,[1] it may still be possible because of numerical issues that θ_i is not in the desired range, and if so, we could reject those samples (not shown in the code).

We can make some comments about the operating range where our approach is reasonable. In general, edge cases should only occur in the tail of the gaussian. Since the standard deviation is β, we are reasonably safe if we assume $\theta_{\max} > 2\beta$. Since the worst case situation is generally $\theta_{\max} \approx \pi/4$, the method above is reasonable for $\beta < \pi/8$, or about $22°$, which is a reasonable range (in practice not much bias is observed for larger roughnesses).

[1]Unfortunately, edge cases are a fact of life when dealing with millions of pixels. Our initial implementation included none of these checks, but tracking down a few strange pixels in test scenes required us to include this edge case detection, and we believe it is a valuable practical addition for JGT readers. Note that no similar concerns arise for ϕ sampling, since that is an angle operated on by trigonometric functions where any wrapping is automatic.

3.4. Inverse CDF for sampling N and final sample direction

We now apply a fairly standard inverse-CDF method for sampling the N term. Recall from equation 4 that $N(\phi) = \cos(\phi/2)$. Note that ϕ lies in the range from $[-\pi, +\pi]$. However, to convert this to a pdf, we need to normalize by a factor of 4. The PDF and CDF are simply

$$\begin{aligned} \mathrm{pdf}(\phi) &= \frac{1}{4}N(\phi) = \frac{\cos(\phi/2)}{4} \qquad\qquad \frac{1}{4}\int_{-\pi}^{\pi} \cos\frac{\phi}{2}\,d\phi = 1 \\ \mathrm{cdf}(\phi) &= \frac{1}{2}\left(1 + \sin\frac{\phi}{2}\right), \end{aligned} \tag{7}$$

where the offset is to ensure the CDF is 0 at $\phi = -\pi$. Inverting this directly gives line 18 in the pseudocode.

We can now go ahead and construct the incident vector or geometric sampling direction (note that the construction is in the hair coordinate system, and therefore somewhat different from the standard spherical coordinates). Finally, we transform this into the appropriate reference frame.

3.5. Sample pdf

Finally, we need to compute the estimator in equation 2, which requires both the value for a sample, as well as the probability distribution function. *Note that if we only need to do BRDF sampling, we need only the final weight (value/pdf), in line 26. The explicit pdf calculations in lines 22- 25 of the pseudocode are only needed for multiple importance sampling.*

First, consider the probability of choosing a given sample direction, separately considering the angles θ_i and ϕ_i. The pdf for ϕ_i is already given above, as $\mathrm{pdf}(\phi_i) = (1/4)\cos\phi/2$. However, the probability for θ_i requires conversion from the probability for θ_h which is what we actually sampled. More generally, we require a probability distribution function,

$$\int_{\theta_i=-\pi/2}^{\pi/2} \int_{\phi_i=-\pi}^{\pi} \mathrm{pdf}(\theta_i, \phi_i)\cos\theta_i\,d\theta_i\,d\phi_i = 1, \tag{8}$$

where the cosine is the solid angle measure in our hair coordinates (compare to the use of a sine for standard spherical coordinates). We have sampled so far not in terms of θ_i but in terms of the half angle θ_h. We also know that $d\theta_i = 2\,d\theta_h$. Putting this together,

$$\begin{aligned} &\int_{\theta_h}\int_{\phi} M(\theta_h)\frac{N(\phi)}{4}\,d\theta_h\,d\phi = 1 \\ \Rightarrow\ &\int_{\theta_i}\int_{\phi_i} \frac{M(\theta_h)}{2\cos\theta_i}\frac{N(\phi)}{4}\cos\theta_i\,d\theta_i\,d\phi_i = 1, \end{aligned} \tag{9}$$

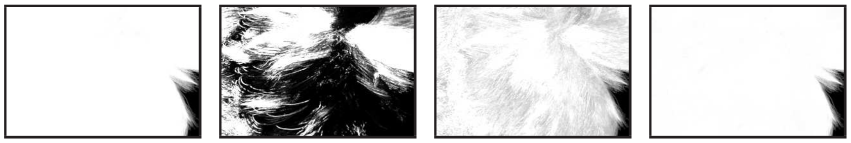

(a) 1 BRDF sample (b) 1 light sample (c) 16 light samples (d) 256 light samples

Figure 2: While our BRDF sampling offers full convergence at 1 sample under a uniform white dome with no shadowing (fig a), light sampling needs all the way to 256 samples (fig d) to resolve the correct normalization. For reference, we show images with 1 and 16 light samples (fig b and c).

where we have switched to integrating over incident angles, and we multiply and divide by $\cos\theta_i$. By inspection from the equation above,

$$\mathrm{pdf}(\theta_i, \phi_i) = \frac{M(\theta_h)N(\phi)}{8\cos\theta_i}, \tag{10}$$

which is directly expressed in line 25 of the pseudocode.[2] Line 24 introduces a neat trick instead of explicitly using the trigonometric function. We know that $\triangle\phi$ is obtained by an inverse sine. Noting that $\cos(\sin^{-1}(u)) = \sqrt{1-u^2}$ and simplifying the algebra, we obtain the result.

3.6. Computing the Estimator and Energy Conservation

Finally, we must compute the sample's contribution to the estimator in equation 2. One condition we would like to ensure is *energy conservation*, that the hair appears uniform when placed in a lighting dome of uniform radiance. This requires the hair BRDF to itself be a probability function, essentially requiring the scattering function to have the same normalization as the pdf. Therefore, we use

$$S(\omega_i, \omega_r) = \frac{M(\theta_h)N(\phi)}{8}, \tag{11}$$

from which it follows that the reflectance-dependent part in equation 2 is given by

$$\frac{S(\omega_i, \omega_r)}{p(\omega_i)} \cdot \frac{\cos\theta_i}{\cos^2\theta_d} = \frac{\cos^2\theta_i}{\cos^2\theta_d}, \tag{12}$$

since all other factors involving M and N cancel. Indeed, this is the beauty of good importance sampling, that most factors cancel, leaving an estimator with

[2]Of course, the factors of 2 and 8 in lines 24 and 25 could easily be pre-cancelled. Other standard optimizations could also be done. These factors have been retained here for readability.

very low variance. We robustly compute (avoiding small values) the cosine denominator $\cos\theta_d = \max(\cos((\theta_i - \theta_r)/2), 1.0e-5)$. In the pseudocode, we also include the overall specular color K_s in the weight.

Our final form in line 26 is even simpler. For sharp specular lobes, incident and reflected cosines will be very similar, as will that of the difference angle. Thus, the right-hand side in equation 12 can simply be replaced with 1. We also note that the original derivation in [Marschner et al. 03] uses a mirror where $\cos\theta_i = \cos\theta_r = \cos\theta_d$, and the rationale for using a denominator with θ_d for rough surfaces, as opposed to θ_i is not clear, except from conditions of reciprocity. Therefore, we directly use the very simple form in line 26, and we have not found this to change the results significantly. Besides simplicity, this formula enforces a form of exact energy conservation; the scattering function is now exactly a probability distribution function (with edge cases handled not with an analytic formula, but implicitly through our earlier discussion; implicitly both the probabilities and value of the scattering function are modified in the same way to give a net Monte Carlo weight of 1).[3]

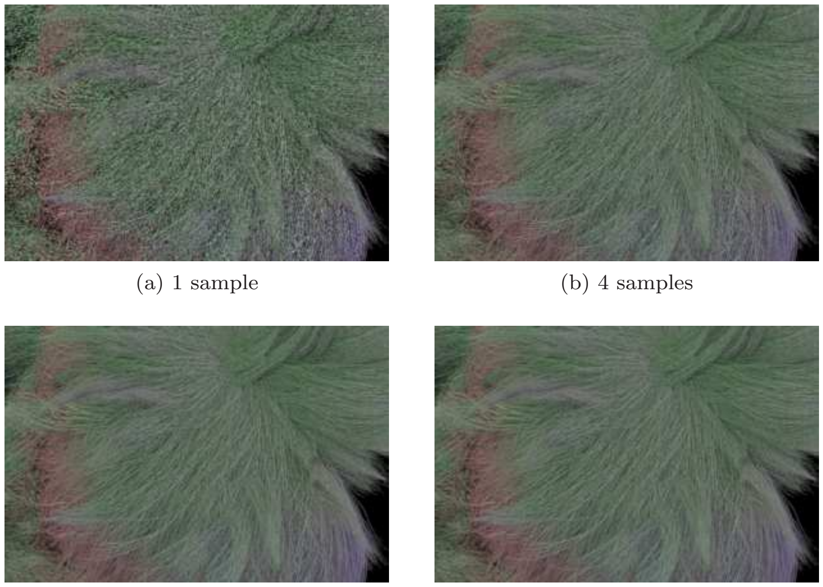
Finally, the overall rendering system will take the weights produced from the BRDF sampler, and multiply them with the lighting for the sample directions, modulated by visibility, and average over all Monte Carlo samples. Note that the overall rendering system cares only about the weight and the BRDF direction. However, we do compute the pdf explicitly, both for instructive purposes, and since it is useful for multiple importance sampling, as discussed next.

Refer to Figure 2, where a common validating "white furnace" test is done. The idea is to make sure that the BRDF correctly integrates to white under a non shadowing uniform white dome. Similarly Figure 3 demonstrates the fast convergence of our approach under arbitrary lighting with full shadowing computations.

4. Multiple Importance Sampling

In practice, the BRDF sampling routine above will often be combined with light sampling in a multiple importance sampling (MIS) framework [Veach and Guibas 95]. One requirement of MIS is that we are able to compute the BRDF value and pdf for an arbitrary direction generated by light sampling. Moreover, there may be cases where we want to use light sampling; we still

[3]Note that the first part of equation 8, and by extension equation 9 requires the gaussian normal distribution function to integrate to 1, which it does over an infinite domain. The integral is approximately 1 over the restricted angular domain, but our computations do not strictly account for the way we handled edge cases to clamp the range of values. This does not create practical problems, especially since the estimator is also set up to compensate and ensure energy conservation, as discussed above.



(a) 1 sample (b) 4 samples

(c) 16 samples (d) ground truth

Figure 3: As can be seen from the figure, BRDF sampling converges rapidly to the true result (ground truth was obtained using light sampling with 256 samples). This example includes full shadow tracing.

need to be able to evaluate the normalized BRDF value in those cases for an arbitrary incident direction. Therefore, we describe the Value and PDF function in algorithm 2, which is largely similar to BRDF sampling.

The main difference is that we now are given as input a list of incident directions (in Dir [ ]). We start with the basic setup as for BRDF sampling, computing θ_r and the constant *denom* term. Instead of computing ϕ_r, we compute the corresponding vector *Rperp* instead (a similar vector *Lperp* will be computed later for incident directions).

We now proceed to compute M, starting by reading in ω_i and determining θ_i. Note that we have ω_i so the cosine can be computed directly without a trigonometric function call. θ_s is now computed directly from the formula (since $\theta_h = \theta_s + \alpha$). From this, we apply the standard formula for M.

For computing N, we compute *Lperp* for the incident direction, just as we calculated *Rperp*. $\cos\phi$ is simply the dot product between these vectors. $\cos(\phi/2)$ is obtained directly from a well-known trigonometric identity, without using any explicit trigonometric functions.

Finally, we need to compute the value and pdf. The pdf is computed just as for the BRDF sampling case, discussed earlier. The value is simply the final weight times the pdf, and we have already seen that the weight is simply K_s.

Algorithm 2. (BRDF Value and PDF for Multiple Importance Sampling)

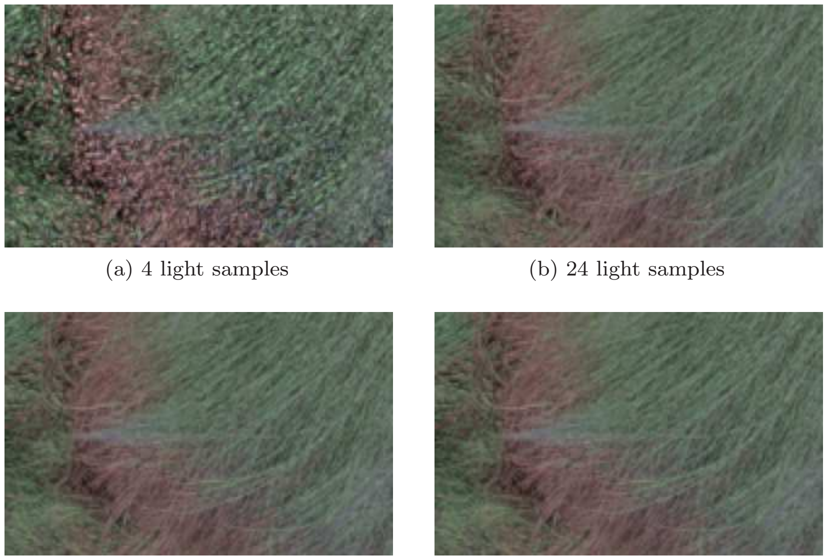
```
public void ValuePDF (vector $\omega_r$; vector Dir [ ]; out BRDFsamp)
    // Basic Setup, compute $\theta_r$ and Rperp to calculate $\phi$
    float $\theta_r = \frac{\pi}{2} - \texttt{acos}(\omega_r[0])$ ;
    vector Rperp = normalize (vector (0.0, $\omega_r[1]$, $\omega_r[2]$) ) ;
    uniform float denom = -0.5 / $\beta$ / $\beta$ ;
    // Now, loop over the required number of samples
    uniform float k ;
    for (k = 0 ; k < numDirections; k += 1) do
        // Compute $M$ term
        vector $\omega_i$ = Dir [k] ;
        float $\theta_i = \frac{\pi}{2} - \texttt{acos}(\omega_i[0])$ ;
        float cosi = sqrt (1.0 - $\omega_i[0]$ * $\omega_i[0]$) ;
        float $\theta_s = (\theta_i + \theta_r)/2 - \alpha$ ;
        float $M = \frac{1}{\beta\sqrt{2\pi}}$ * exp ($\theta_s$ * $\theta_s$ * denom) ;
        // Compute $N$ term
        vector Lperp = normalize (vector (0.0, $\omega_i[1]$, $\omega_i[2]$) ) ;
        // Trig identity $\cos\phi = 2\cos^2(\phi/2) - 1$
        float $N$ = sqrt ( (1.0 + Rperp $\cdot$ Lperp ) * 0.5 ) ;
        // Compute Value and PDF
        BRDFsamp $\rightarrow$ pdf[k] = $M$ * $N$ / (8.0 * cosi) ;
        BRDFsamp $\rightarrow$ value[k] = $K_s$ * BRDFsamp $\rightarrow$ pdf[k] ;
        // If we desire to keep the $\cos\theta_d$ term, we can multiply
           this by $(\cos\theta_i/\cos\theta_d)^2$.
    end
```

Figure 4 compares light sampling with BRDF sampling and MIS. In this environment, the advantage of MIS is rather minimal. However, as is common with MIS, situations with more isolated light sources would demonstrate benefits from light sampling.

References

[Acklam 03] P. Acklam. "An Algorithm for Computing the Inverse Normal Cumulative Distribution Function." Available online, http://home.online.no/˜pjacklam/notes/invnorm/, 2003.

[Box and Muller 58] G. Box and M. Muller. "A Note on the Generation



(a) 4 light samples (b) 24 light samples

(c) 4 BRDF samples (d) MIS 4 light and 4 BRDF samples

Figure 4: Here we compare the quality of BRDF sampling versus light sampling. Under this environment, 4 BRDF samples (fig c) are equivalent to about 24 light samples (fig b). We also provide (fig d) an MIS render (a combination of light 4 samples and 4 BRDF samples): the differences are rather subtle since light sampling is much worse than BRDF sampling, but if you look closely, you can see better definition of the individual strands.

of Random Normal Deviates." *The Annals of Mathematical Statistics* 29:2 (1958), 610–611. Available online (http://en.wikipedia.org/wiki/Box-Muller_transform).

[d'Eon et al. 11] E. d'Eon, G. Francois, M. Hill, J. Letteri, and J. Aubry. "An Energy-Conserving Hair Reflectance Model." *Computer Graphics Forum (EGSR 11)* 30:4 (2011), 1181–1187.

[Goldman 97] D. Goldman. "Fake Fur Rendering." In *SIGGRAPH 97*, pp. 127–134, 1997.

[Kajiya and Kay 89] J Kajiya and T Kay. "Rendering Fur with Three Dimensional Textures." In *SIGGRAPH 89*, pp. 271–280, 1989.

[Lawrence et al. 04] J Lawrence, S Rusinkiewicz, and R Ramamoorthi. "Efficient BRDF Importance Sampling Using a Factored Representation." *ACM Transactions on Graphics (SIGGRAPH 2004)* 23:3.

[Lokovic and Veach 00] T Lokovic and E Veach. "Deep Shadow Maps." In *SIGGRAPH 00*, pp. 385–392, 2000.

[Marschner et al. 03] S. Marschner, H. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. "Light Scattering from Human Hair Fibers." *ACM Transactions on Graphics (SIGGRAPH 03)* 22:3 (2003), 780–791.

[Sadeghi et al. 10] I. Sadeghi, H. Pritchett, H. Jensen, and R. Tamstorf. "An artist friendly hair shading system." *ACM Transactions on Graphics (SIGGRAPH 10)* 29:4.

[Veach and Guibas 95] E Veach and L Guibas. "Optimally Combining Sampling Techniques for Monte Carlo Rendering." In *SIGGRAPH 95*, pp. 419–428, 1995.

[Xie et al. 07] F. Xie, E. Tabellion, and A. Pearce. "Soft Shadows by Ray Tracing Multilayer Transparent Shadow Maps." In *EuroGraphics Symposium on Rendering*, pp. 265–276, 2007.