# Importance Sampling for Production Rendering

Mark Colbert
Google
ImageMovers Digital

Simon Premože

Guillaume François
Weta Digital
Moving Picture Company

SIGGRAPH 2010 Course

## Abstract

Importance sampling provides a practical, production-proven method for integrating diffuse and glossy surface reflections with arbitrary image-based environment or area lighting constructs. Here, functions are evaluated at random points across a domain to produce an estimate of an integral. When using a large number of sample points, the method produces a very accurate result of the integral and provides a strong basis for simulating complex problems such as light transport.

Frequently, using the necessary number of samples to reach the exact result is too computationally expensive and fewer samples are evaluated at the cost of visual noise, or variance, within the image. Importance sampling offers a means to reduce the variance by skewing the samples toward regions of the illumination integral that provide the most energy. For instance, the direction of specular reflection or a bright light source within an environment more likely represent the final value of the integral than a random sample.

The variance can be reduced more efficiently by combining multiple components of the illumination integral, such as the lighting and material function, to determine where to sample, which is the principle of Multiple Importance Sampling (MIS).

As an alternative to the noise in importance sampling, Filtered Importance Sampling (FIS) can provide fast integration, where the lighting environment look-ups are pre-filtered to give a smoother result with a significantly smaller number of samples.

Importance sampling, MIS and FIS have various practical implications. In this quarter-day course, we cover the necessary background for using Monte Carlo-based techniques for direct lighting and explain how various visual effects companies use these shading methods in their production pipelines.

# Contents

## Course Overview

**Format:** Quarter-Day (1.75 hr) Course Outline

- Introduction **(1 minute)** *Mark Colbert*
- Background **(20 minutes)**
  - Overview of the Illumination Integral and Monte Carlo Rendering
  - Importance Sampling
  - Phong BRDF Sampling
  - Exponential Distribution Sampling
  - Quasi-Random Low-Discrepancy Sequences
- Filtered Importance Sampling (FIS) **(15 minutes)**
  - Motivation for a Smoother but Biased Estimator
  - BRDF Proportional Filtered Importance Sampling with Image-based Environment Lights
    - MIP-map Filtering
    - Ptex-based Cube Map Filtering
  - Analysis and Convergence
- FIS for Area Lights in "A Christmas Carol" **(10 minutes)**
  - Decoupled Lighting Model
  - Ray Differentials
  - Filter Selection
  - Slide Map
  - Aliasing Artifacts
  - Real-time Interactive Setup for Lighters
- Importance Sampling Framework at MPC **(25 minutes)** *Guillaume François*
  - Image-based Environment Importance Sampling
  - Image-based Lighting Constraints
    - Energy Conserving BRDFs and Albedo Pump-up
    - Dealing with Number of Samples
    - Raytracing strategies in "Clash of the Titans"
  - Quick Implementation of Iridescence with Filtered Importance Sampling
  - Stochastic versus Quasi-Random Sequences
- Multiple Importance Sampling (MIS) **(30 minutes)** *Simon Premože*
  - Overview of MIS
    - Mixture Sampling
    - Control Variates
    - Defensive Importance Sampling
    - Mixture Sampling
  - Practical MIS
    - Sampling Strategies
    - Correlated Sampling
  - FIS and MIS
- Questions **(5 minutes)** *All Presenters*

## Biographies

### Mark Colbert

Software Engineer
Google
mark.c.colbert@gmail.com
http://www.cs.ucf.edu/~colbert

Mark is a Software Engineer at Google working within on the Street View project. He was a Research and Development Engineer at ImageMovers Digital where he worked in the render development group writing shaders and pipeline tools. He was the lead developer for the real-time lighting pre-visualization tool chain. His film credits include A Christmas Carol and the upcoming movie Mars Needs Moms. In 2008, he graduated with a Ph.D. in Computer Science from the University Central Florida.

### Simon Premože

simon 'DOT' premoze 'AT' gmail 'DOT' com

Simon Premoze received a B.S. in Computer Science from the University of Colorado at Boulder and was granted a Ph.D. in Computer Science from the University of Utah. He was a postdoctoral research associate at Columbia University where he worked on volume rendering and global illumination. He has been an R&D engineer at Industrial Light and Magic where he worked on a variety of rendering problems in production. His current research interests include global illumination and rendering algorithms, modeling natural phenomena and reflectance models. Previously he worked on computer simulation and visualization of liquid crystal phase transitions and dynamics of liquid crystals.

### Guillaume François

Shader Writer
Weta Digital
guillaume.francois@gmail.com

Guillaume Francois was a Shader Writer at the Moving Picture Company where he is responsible for the Image Based Lighting shading pipeline. His interests and work also include atmospheric scattering, skin modeling and rendering and other rendering challenges. His film credits include G.I. Joe: The Rise of Cobra, Prince of Persia: The Sands of Time and Clash of the Titans. In 2008, he graduated with a Ph.D. in Computer Science from the University of Rennes I.

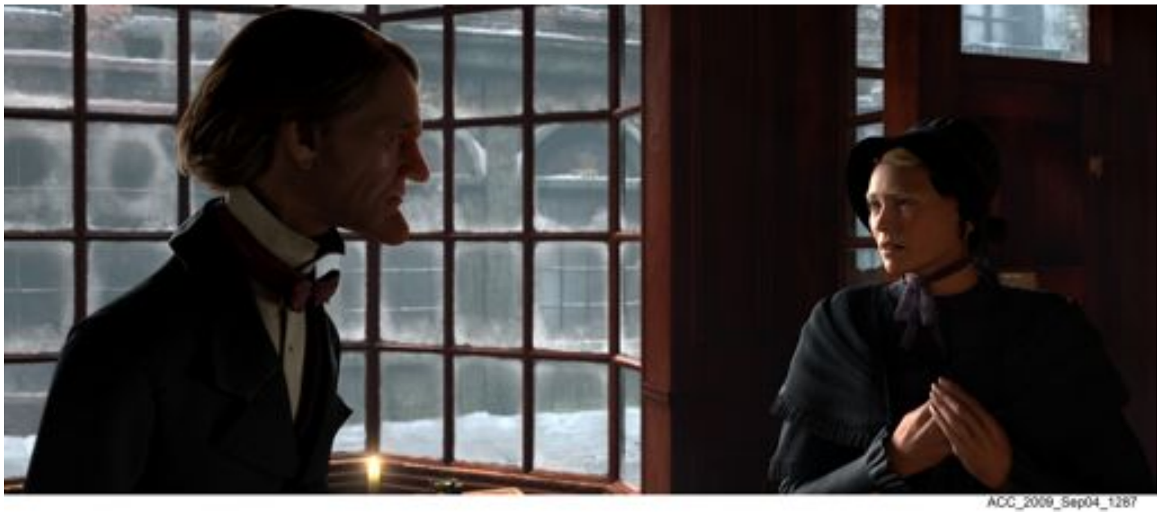


Figure 1: The broad glossy and diffuse reflections from an area light source were required to simulate the soft appearance of Belle in Disney's A Christmas Carol. Here, filtered importance sampling was used to provide an efficient and effective result for the soft reflections.

## 1 Introduction

As the visual effects and game industries strive for increasing realism, more complex lighting and material models are being tested and applied to achieve high fidelity images. As one approach, area and image-based lighting models are being combined with diffuse and glossy surface models to provide rich reflection detail (Figure 1). Unfortunately, the computation time required for brute force integration of these models is rather expensive. The cost can have a strong impact on the effectiveness and benefit of complex lighting models in a production pipeline. For visual effects, this implies the iteration time expands as the render times increase for an artist tweaking the appearance of a synthesized image. For games, the large computation may drop game performance below an acceptable level preventing the use of the model all together.

*Importance sampling* provides a means to reduce the cost of brute force integration by selectively evaluating elements of the integrand based on prior knowledge, i.e. an educated guess. However, determining the prior knowledge is tricky since the illumination integral contains the product of the incoming light and the material reflectance function. Methods such as *Multiple Importance Sampling* (MIS) provide a way to account for the various components and solve for the reflectance with a relatively small number of samples.

Another method of reducing this computational burden is through *Filtered Importance Sampling* (FIS) [CK07, KC08]. Although the method was originally intended for GPU-based material design, the algorithm's flexibility and simple parallelization makes it amenable for rendering directly illuminated glossy surfaces in a production context. To that end, the approach has been used in movies such as Iron Man 2, A Christmas Carol, Terminator 4, Clash of the Titans, and Percy Jackson and the Olympians: The Lightning Thief.

The goal of this course is to provide the audience with the necessary nuts and bolts in theory and code to understand and implement an importance sampling-based shading system. To that end, the following is presented as a tutorial for the reader.

### 1.1 Overview of Course Material

In the following course notes, we discuss *Monte Carlo* (MC) numerical integration and how importance sampling can provide a computationally cheaper solution than brute force MC (Section 2). We overview the underlying theory and math for integrating area and image-based lighting environments with material reflectance models (Section 3) and apply the importance sampling theory to rendering (Section 4). In the process, the formula and code for material-based and image-based environment sampling will be derived as reference for the reader.

Next, we explain how the importance sampling framework can be adapted to support filtering and demonstrate how to improve computational efficiency for area and image-based environment lighting with FIS (Section 5). We also provide empirical analysis on the various artifacts associated with FIS.

We discuss other means to reduce the cost associated with importance sampling by using a technique called *Multiple Importance Sampling* (Section 6). Last, practical notes are provided for implementing importance sampling in a production pipeline (Section 7). This includes describing some solutions and tricks to efficiently render large and complex scenes when using the techniques presented in this course (Section 8).

## 2 Monte Carlo Methods

The term *Monte Carlo* refers to all methods that use a statistical sampling processes to approximate solutions to quantitative problems. It can be used for a wide variety of probabilistic problems ranging from numerical integration to optimization. These methods are used in many application domains such as economics, robotics and nuclear engineering.

In this section, we describe some basic concepts of Monte Carlo integration. After a brief overview of the Monte Carlo methods, we introduce the principle of Monte Carlo integration and look at some of the basic statistical properties. Then, we describe some basic variance reduction techniques, such as importance sampling, control variates, and mixture sampling, that we use in later sections in the context of light transport. This section is only a brief summary, but it does provide some insights and intuition about why some methods perform better than other and describes the circumstances one should use a particular method. We encourage interested readers to learn more about Monte Carlo methods and probability in many excellent books [KW86, SG69, HH64] and papers that exist on the topic.

Readers who are mostly interested in the practical implementation Monte Carlo methods for rendering can skip this section.

### 2.1 Estimators

A *continuous random variable* $X$ is a quantity that randomly takes on a value $x$ that lies on the real line $(-\infty, \infty)$. The values of $x$ can be quantitatively described by the *probability density function* (PDF) $p$. The probability that $x$ will take a value on some interval between $a$ and $b$ is then

$$Pr\{a \leq X \leq b\} = \int_a^b p(x)dx. \tag{1}$$

The probability density function $p(x)$ must satisfy two conditions:

1. It is always positive:

$$p(x) \geq 0$$

2. It is normalized:
$$\int_{-\infty}^{\infty} p(x)dx = 1$$

It is important to understand the difference between *probability* and the probability density function. The probability, or likelihood, of an event takes values strictly between 0 (*impossible event*, it never happens) and 1 (*certain event*, it always occurs). On the other hand, the probability density function describes the relative likelihood of a random variable (or event) having a certain value. For instance, if $p(x_1) = 10$ and $p(x_2) = 100$, then the random variable with the PDF $p$ is ten times more likely to have a value near $x_1$ than near $x_2$. The relationship between the probability density function $p$ and the probability $Pr$ is defined in Equation (1).

Other important concepts to understand are expected value and variance of a random variable. The *expectation* (or *expected value*) of a random variable $Y = f(X)$ is
$$E[Y] = \int_{\Omega} f(x)p(x)dx \tag{2}$$
and its variance is
$$V[Y] = E[(Y - E[Y])^2]. \tag{3}$$
Intuitively, expected value (or *mean value*) is just the average value of the random variable. Note that expected value should not be confused with the most probable value. On the other hand, variance measures how much the values of some random variable deviate from its mean or expected value. The higher the variance, the more values differ from the average value.

The expected value has a few useful properties:

1. The expected value of the sum of two random variables $X$ and $Y$ is the sum of the expected values of those variables:
$$E[X + Y] = E[X] + E[Y]$$
Since functions of random variables are also random variables, this principle applies to the sum of functions of random variables:
$$E[f(X) + g(Y)] = E[f(X)] + E[g(Y)]$$
The above holds true even if variables $X$ and $Y$ are correlated.
2. For any constant $a$, the expected value and variance for $aX$ are
$$\begin{aligned} E[aY] &= aE[Y] \\ V[aY] &= a^2V[Y] \end{aligned}$$

If we want to compute an approximation to some unknown quantity $Q$ (*i.e.*the *estimand* or quantity of interest), a function $F$ of random variables $X_1, \ldots, X_N$ is called an *estimator* if its mean (expected value) $E[F]$ is a usable approximation to $Q$:
$$F_N = F_N(X_1, \ldots, X_N) \tag{4}$$
A particular numerical value of $F_N$ is called an *estimate*. $Q$ can be any function that we might be interested in. In rendering, $Q$ can be the amount of light that reaches a point on a surface or the amount of light reflected from the surface.

There are many possible estimators. In general, we want Monte Carlo estimators to provide good estimates as fast as possible. **How do we choose a good estimator?** First, we need to establish some criteria for what *good* means by looking at the properties of Monte Carlo estimators:

- **Error**

$$\text{error} = F_N - Q$$

  Mean Square Error (MSE) of an estimator $F$ is then

$$\text{MSE} = E[(F_N - Q)^2] \tag{5}$$

- **Bias**
  Bias $\beta$ is the expected value of the error:

$$\beta[F_N] = E[F_N - Q] \tag{6}$$

  The estimator is *unbiased* if $\beta[F_N] = 0$ for sample size $N$:

$$E[F_N] = Q \qquad \text{for all} N \geq 1. \tag{7}$$

  An obvious advantage of the unbiased estimator is that we are guaranteed to get the correct value of quantity of interest $Q$ if enough samples are taken. Furthermore, the expected value of an unbiased estimator will be the correct value after any number of samples. The mean square error of the estimator can be also written as

$$\text{MSE}[F_N] = V[F_N] + \beta[F_N]^2. \tag{8}$$

  For unbiased estimators, the MSE is the same as the variance. For biased estimators, the error is much more difficult to estimate. It is also important to know that a biased estimator may not give a correct estimate for $Q$ even if an infinite number of samples are taken. In practice, a biased estimator may have some desirable properties, such as lower variance, which makes it very appealing for use in computer graphics. For example, in rendering, noise is a manifestation of variance. While taking more samples reduces the amount of noise, rendering using a biased estimator may have less noise for the same number of samples albeit producing different images.
- **Consistency**
  An estimator is *consistent* if the error goes to zero as the number of samples grows:

$$Pr\{\lim_{N\to\infty} F_N = Q\} = 1. \tag{9}$$

  The above equation is essentially saying that if we use a consistent estimator, we are one hundred percent certain that the answer is correct if we increase the number of samples. Consistency is a stronger condition than requiring the estimator to be unbiased. It is still possible that an unbiased estimator is not consistent, in which case its variance is infinite. A biased estimator is consistent if its bias $\beta$ decreases to 0 as the number of samples $N$ increases.

## 2.2 Monte Carlo Integration

The basic idea behind Monte Carlo integration is evaluation of the integral

$$I = \int_\Omega f(x)dx \tag{10}$$

using random sampling. Here, $N$ random points $X_1, X_2, \ldots, X_N$ are independently sampled from some density function $p$ and used to approximate $I$,

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i). \tag{11}$$

**Notation note:** A realization of an estimator $F$, namely $F_N$, is the same as $\hat{I}_N$. The subscript $N$ emphasizes that $\hat{I}$ is still a random variable and therefore its properties depend on how many samples were chosen.

As $N$ increases, the expected error of this estimate decreases. We want to choose $N$ such that we have confidence that the estimate $\hat{I}_N$ is good. The estimator $\hat{I}_N$ is a crude but unbiased estimator for $I$ and its variance is

$$V[\hat{I}_N] = V[\frac{1}{N}\sum_{i=1}^{N} f(X_i)] = \frac{1}{N}V[f(X)]. \tag{12}$$

From this variance estimate $V[\hat{I}_N]$ we can conclude the following:

1. The standard error of the estimator decreases with the square root of the sample size $N$. Recall that the standard error of $\hat{I}_N$ is $V[\hat{I}_N]^2$, so while the variance of the estimate is proportional to $1/N$ the standard deviation is proportional to $1/\sqrt{N}$. Therefore, to reduce the error in half, we have to quadruple the number of samples.
2. The statistical error is independent of the dimensionality of the integral. This simply means that the computation does not increase exponentially when the dimensionality of the integral increases.

So far, we have not made any assumptions about function $f(x)$ we are trying to integrate. On the other hand, in the above discussion we have assumed that our random variable $X$ is uniformly distributed over the integration domain $\Omega$. Loosely speaking, a uniform distribution implies that the probability of choosing each sample is equal. Unfortunately, real problems are rarely this simple. For example, function $f(x)$ can be zero in many regions and have very high values in other. If uniformly sampling the domain $\Omega$, we may get very large variance. Also, sometimes it may not be possible to sample a space uniformly. In order to alleviate these problems, we can rewrite the estimator from Equation (11) as

$$\begin{aligned} I &= \int_\Omega f(x)dx \\ &= \int_\Omega \frac{f(x)}{p(x)}p(x)dx, \end{aligned}$$

where $p(x)$ is a probability density function in $\Omega$. We can now generate $N$ samples from distribution $p(x)$ (instead of uniformly sampling $\Omega$) to get

$$\hat{I}_p = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}. \tag{13}$$

The simple Monte Carlo estimator was saw in Equation (11) is just a special case of the more general estimator in Equation (13) with $p(x)$ being a uniform distribution in $\Omega$. This estimator has the same variance properties we have seen above.

One major advantage of Monte Carlo integration is that it is easy to understand and simple to use. If we can generate random samples using some density $p(x)$ and have the ability to compute the sample weights, $w_i = \frac{f(X_i)}{p(X_i)}, i = 1, \ldots, N$, then we can evaluate the integral. Monte Carlo methods are also flexible, robust and work well in higher dimensions where other numerical methods might fail.

## 2.3 Variance Reduction Techniques

One of the biggest disadvantages of Monte Carlo methods is a relatively slow convergence rate. As we have already discussed above, the root mean square (RMS) error converges slowly at a rate of $O(1/\sqrt{N})$, so we need to quadruple number of samples $N$ to halve the error.

Ideally, we would like to use an estimator which has both small variance and is computationally efficient. *Efficiency* of a Monte Carlo estimator $F$ is

$$\epsilon[F] = \frac{1}{V[F]T[F]} \tag{14}$$

where $V[F]$ is the variance and $T[F]$ is the time needed to evaluate $F$. Therefore, the more efficient the estimator is the lower the variance in a given (fixed) amount of time.

One of the fundamental goals in researching Monte Carlo methods is to find or design efficient estimators. These techniques are often called *variance reduction techniques* and include *importance sampling*, *control variates*, and *adaptive sampling*. We briefly review some of these techniques. In later sections, we apply these techniques to the direct illumination rendering problem.

### 2.3.1 Importance Sampling

Recall that a Monte Carlo estimator for some function $f(x)$ over domain $\Omega$ is

$$I = \int_\Omega \frac{f(x)}{p(x)} p(x) dx$$

and the estimator is

$$\hat{I}_p = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}.$$

The variance of the estimator $\hat{I}_p$ depends on the density $p(x)$ from which random samples are drawn. If we choose the density $p(x)$ intelligently, the variance of the estimator is reduced. This is called *importance sampling*. $p(x)$ is called the *importance density* and $w_i = \frac{f(X_i)}{p(X_i)}$ is the *importance weight*.

The best possible sampling density is $p^*(x) = cf(x)$ where $c$ is proportionality constant

$$c = \frac{1}{\int_\Omega f(x)dx}. \tag{15}$$

Here, the constant ensures that $p^*$ is normalized (*i.e.*, it integrates to 1). The density $p^*(x)$ yields an estimator with zero variance. In practice, we cannot use this density, because we must know the value of the integral we want to compute to evaluate $c$. However, if we choose an importance density $p(x)$ that has a similar shape to $f(x)$, the variance can be reduced. It is also important to choose an importance density $p$ such that it is simple and efficient to evaluate. In practice, $p$ can be designed by doing some of the following:

1. Discard or approximate some parts of $f(x)$ such that function $g(x) = f(x)p(x)$ can be integrated analytically.
2. Construct a low dimensional discrete approximation of $f(x)$.
3. Approximate $f(x)$ by using Taylor expansion.

After $g(x)$ is designed with any of the above methods, the density is then set to $p(x) \propto g(x)$. We show in next sections how to choose and compute densities in practice.

**Note:** If the sampling density is not chosen carefully, the variance can be increased and can actually be infinite. Importance sampling is very effective when function $f(x)$ has large values on small portions of the domain. Another common problem that happens in importance sampling is when the sampling density has a similar shape to $f(x)$ except that $f(x)$ has longer (wider) tails. In this case, the variance can become infinite. While importance sampling is a useful and powerful technique it should be used with care. Inappropriate importance density can result in poor estimates of the integral.

### 2.3.2 Stratified Sampling

If we partition integration domain $\Omega$ into a set of $m$ disjoint subspaces $\Omega_1, \ldots, \Omega_m$ (*strata*), we can evaluate the integral as a sum of integrals over the stratum $\Omega_i$. If we generate $n_i$ samples in each stratum (subspace $\Omega_i$), the estimator becomes

$$\hat{I} = \sum_{i=1}^{m} \frac{1}{n_i} \sum_{k=1}^{n_i} f(X_{i,k}) \tag{16}$$

whose variance is

$$V(\hat{I}) = \sum_{i=1}^{m} \frac{V_i}{n_i} \tag{17}$$

where $V_i$ is the variance of $f(x)$ in stratum $\Omega_i$. The expected error of this method, *stratified sampling*, is never higher than variance of ordinary unstratified sampling [Mit96]. However, stratified sampling is often better than importance sampling. The two methods can be combined to lower variance even further. Stratified sampling works well for low-dimensional integration, but it does not scale well for integrals of high dimensionality. The number of samples must also be chosen such that there is at least one sample drawn from each stratum.

### 2.3.3 Control variates

If we can rewrite the estimator as

$$I = \int_{\Omega} g(x)dx + \int_{\Omega} (f(x) - g(x))dx \tag{18}$$

where function $g(x)$ can be analytically integrated and has the following property:

$$V[f(x) - g(x)] \leq V[f(x)] \tag{19}$$

then a new estimator is

$$F = \int_{\Omega} g(x)dx + \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i) - g(X_i)}{p(X_i)}. \tag{20}$$

The variance of this new estimator will be lower than the original estimator.

**How do we decide whether to use importance sampling or control variates?** Given function $g(x)$ that is an approximation of $f(x)$, then $g(x)$ can be used either as an importance density or a control variate. If $f(x) - g(x)$ is approximately uniform (constant), then using $g(x)$ as a control variate is more efficient. If $f(x)/g(x)$ is approximately constant, then using importance sampling is more efficient. Note that if $g$ is proportional to $p$ then the two estimators differ only by a constant, and have therefore the same variance. If $g$ is already used as the importance density, it would not be useful as a control variate, because the variance would not be reduced. Another criteria for choosing between importance sampling and control variates is whether $g(x)$ can be integrated analytically (control variates may be preferable) or $g$ can be sampled analytically (importance sampling).

### 2.3.4 Defensive importance sampling

We already mentioned in Section 2.3.1 that even if a sampling density $p(x)$ has roughly the same shape as a target function $f(x)$, but $f(x)$ has longer tails, importance sampling will fail. When we draw a sample from the tails of $p(x)$, the importance weight can be many times larger than weights in other parts of $p(x)$. This causes high variance and in extreme cases the variance can be infinite. This deficiency can be address with *defensive importance sampling* [Hes95] which uses a *defensive mixture distribution* $p_\alpha(x)$ instead of only the density $p(x)$:

$$p_\alpha(x) = \alpha q(x) + (1 - \alpha)p(x). \tag{21}$$

Here, $0 < \alpha < 1$ and $q(x)$ is the target distribution. If we want to compute the integral

$$I = \int_\Omega f(x)q(x)dx \tag{22}$$

where $q(x)$ is the target density on the integration domain. The defensive mixture distribution $p_\alpha(x)$ guarantees that the variance is bounded by $1/\alpha$ times the variance of the uniform distribution estimator. It also bounds the importance weight to $1/\alpha$. However, oftentimes it may not be easy or possible to sample from the target density $q(x)$. If $q(x)$ can be decomposed into a product of several (simpler) densities, $q(x) = q_1(x), \ldots, q_n(x)$ and each $q_i(x)$ can be easily sampled, then a more general mixture distribution of $m$ densities can be used:

$$p_\alpha = \alpha_0 p(x) + \sum_{j=1}^{m} \alpha_j q_j(x). \tag{23}$$

Here, the sum of all weights $\alpha_j$ is one and each weight is greater than zero.

### 2.3.5 Multiple Importance Sampling

Many times we have to integrate a complex function whose target distribution has multiple modes (peaks or bright regions) and sampling with a single importance density may not capture all regions of the integrand. For example, this is a very common problem in rendering. If our scene contains diffuse and glossy surfaces illuminated by small and large area lights, we face a difficult decision about what sampling strategy to use. For diffuse surfaces, one sampling strategy might be preferable to another. However, the opposite might be true for glossy surfaces.

Suppose that we have $n$ different densities, $p_1(x), \ldots, p_n(x)$, and generate $n_i$ samples for each $p_i(x)$. Now, we have many different sampling strategies that work well in different regions of the integrand, but are not good over the entire domain. The question is how to combine multiple strategies that minimize the overall variance without introducing bias. As a naïve approach, one could average the sampling strategies. However, this will not produce optimal results [VG95].

Instead, we combine all $n$ sampling strategies giving the estimator

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{24}$$

where weight $w_i$, $w_1, \ldots, w_n$, provide weight for each sample drawn from some sampling strategy $p_i$. All weights must be non-zero and the total sum must be 1 to ensure that the estimator remains unbiased. An obvious weighting function would be

$$w_i(x) = c_i \frac{p_i(x)}{q(x)} \tag{25}$$

where

$$q(x) = c_1 p_1(x) + \cdots + c_k p_k(x) \tag{26}$$

and all coefficients $c_i$ are nonzero and sum to 1.

In general, the best choice of weights turns out to be

$$w_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)}. \tag{27}$$

If we take exactly one sample, $n_i = 1$ from each sampling density, the weight $w_i$ will be set according to current sampling strategy at $x$ compared to the rest of the strategies:

$$w_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)}. \tag{28}$$

This weighting strategy is called the *balance heuristic* and is nearly optimal. It is possible to design better strategies for special cases, but universally the balance heuristic out performs most other stratigies.

**What is the difference between MIS and Defensive Importance Sampling?** Multiple importance sampling (MIS) is optimal for a given set of sampling strategies. However, if we have chosen a bad or inadequate sampling strategy, MIS will not reduce variance. For example, if one of the strategies takes too many samples from low-valued regions and not enough from high-valued regions, the variance will increase. On the other hand, defensive importance sampling (DIS) can improve variance when a sampling strategy $p$ is inadequate. When combined with a uniform density, DIS guarantees that the integrand will be sampled over entire domain. MIS with balance heuristic can be viewed as a special case of DIS (see Equation (21), and factor $\alpha$ coming from balance heuristic weights).

We will show in later sections how to use MIS in practice and how to design sampling strategies for rendering.

## 3 Direct Illumination Formulation

The goal in direct illumination rendering is to compute for each shading point how much light from the surrounding environment is reflected toward the virtual camera (Figure 2). To convert the incoming light from one particular direction and position, $L_i(\omega_i, \mathbf{x})$, into reflected light toward the direction of the camera, $\omega_o$, we use the material function $f$, known as the *Bidirectional Reflectance Distribution Function* (BRDF). Common BRDFs are the the Lambert [Lam60] and Cook-Torrance [CT82] reflection models that resemble the `diffuse` and `specular` functions in Pixar's RenderMan. To compute the total reflected light, $L_o(\omega_o, \mathbf{x})$, the contributions from every incident direction $\omega_i$ must be summed (or integrated, in the limit) over the hemisphere $\mathcal{H}$,

$$L_o(\omega_o, \mathbf{x}) = \int_\Omega L_i(\omega_i, \mathbf{x}) f(\omega_o, \omega_i) \cos\theta_i d\omega_i. \tag{29}$$

This equation is referred to as the *reflectance equation* or the *illumination integral*. Here, $\theta_i$ is the angle between the surface normal and the incoming light direction $\omega_i$.

In image-based lighting, the incoming light $L_i$ is approximated by an environment map, where each pixel corresponds to an incoming direction $\omega_i$ and occlusion of the light can be computed using techniques such as ray tracing or shadow mapping. Even if ignoring occlusion, the numerical integration of Equation (29) for one shading point

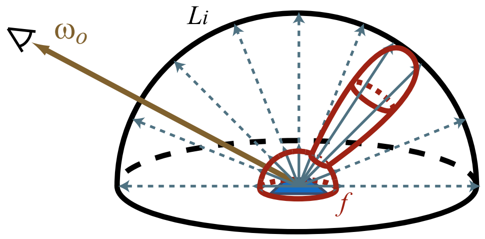


Figure 2: Components of the Illumination Integral. Here, the incoming hemisphere of light rays are multiplied by the BRDF $f$ and reflected towards the camera $\omega_o$ at a given shading point.

in the image requires thousands of pixels in the environment map to be multiplied with the BRDF and summed. Similarly when using an area light, the light's surface can be discretized into many small elements. Each element is considered a point light source whose contribution is subtended by the orientation and distance of the element to the shading point. The elements' emissions are attenuated by the BRDF and summed together to estimate the original area light's reflection. In both cases, these operations are too computationally expensive for real-time rendering and often too slow for offline rendering.

Since we cannot afford evaluating the illumination integral for all incident directions, we randomly choose a number of directions and evaluate the integrand, $L_i(\omega_i, \mathbf{x}) f(\omega_o, wi) \cos\theta_i$, at these directions to obtain *samples* of the integral. The average of these samples produces an estimate of the integral, which is the essence of Monte Carlo quadrature discussed in Section 2. If we had an infinite set of samples, the average of the samples would equal the true value of the integral. This is denoted as the *expected value* of the estimator. However, using only a practical, finite set of samples, the estimate varies from the actual solution and introduces noise, or *variance*, in the image. One way to reduce this noise is by choosing the most important samples that best approximate the integral.

## 4 Importance Sampling for Direct Illumination

Generating uniform random directions is not the best approach for approximating the illumination integral, Equation (29), if we have a rough idea about the behavior of the integrand. For instance, if we are reflecting an environment on a glossy material, it seems intuitively more effective to sample directions around the specular direction (i.e. the mirror reflection direction), since much of the reflected light originates from these directions. As discussed in Section 2, we represent this mathematically using a *probability density function* (PDF) that defines the optimal directions for sampling. Recall, the PDF is a normalized function, where the integration over the entire domain of the function is equal to 1, and the peaks represent important regions for sampling.

However, by skewing sample directions, not all estimates of the integral are equal and thus we must weigh them accordingly when averaging all the samples. For instance, one sample in the trough of the PDF is representative of what would be many samples if uniform sampling was used. Similarly, one sample around the peak of a PDF represents only a few samples with uniform sampling. To compensate for this property of the PDF-proportional sampling, we multiply each sample by the inverse of the PDF. This

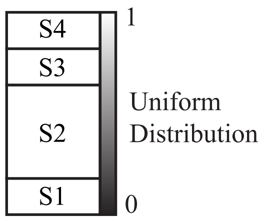


Figure 3: PDF proportional mapping. In this case, if a random number is chosen, where there exists an equal likelihood that any value between 0 and 1 will be produced, then more numbers will map to the important sample S2 and thus that direction will be sampled more often.

yields a Monte Carlo estimator that uses the weighted average of all the samples,

$$L_o(\omega_o, \mathbf{x}) \approx L'_o(\omega_o, \mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(\omega_i, \mathbf{x}) f(\omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}. \tag{30}$$

Defining an optimal PDF for the illumination integral requires an accurate approximation of the product between the material BRDF and the incident lighting. In practice, if the BRDF has higher frequencies than the environment, such as a glossy surface reflection with a blurry environment light source, the BRDF alone is sufficient to sample. Conversely, if the BRDF is low frequency, such as with diffuse reflection, then the environment is ideal to use for sampling. In the following sections, we will cover how to sample solely from the BRDF and solely from an image-based environment. Sampling from both distributions simultaneously is discussed in Section 2.3.5.

## 4.1 BRDF-based Importance Sampling

As one of the components of the illumination integral, the BRDF can provide a strong basis to guide the importance sampling for glossy surface reflection. While various ways exist to create uniformly random samples, such as pseudo-random or quasi-random numbers generators, we must convert the uniformly random values to be proportional or nearly proportional to the BRDF. Two BRDFs discussed and commonly used in production for glossy surface reflection include Phong's reflectance model [Pho75] and the Cook-Torrance model [CT82] (similar to `specular` in Pixar's RenderMan). Since the Cook-Torrance model is predominately driven by an exponential distribution, we will look at a general sampling strategy for this distribution. Here, we ignore other components of the reflectance model, such as the Fresnel term, since this can be handled using techniques such as multiple importance sampling (Section 2.3.5). The following provides a full derivation of these sampling formulas as reference for the reader.

### 4.1.1 Inverse Transform Sampling

As one method for warping the uniform distribution, we can convert the PDF into a *Cumulative Distribution Function* (CDF). Intuitively, think of a CDF as a mapping between a uniform distribution to a PDF-proportional distribution. In the discrete case, where there are only a finite number of samples, we can define the CDF by stacking each sample. For instance, if we divide all possible sampling directions for rendering into 4 discrete directions, where one of the sample directions, S2, is known a priori to be more important, then the probabilities of the 4 samples can be stacked together as depicted in Figure 3.

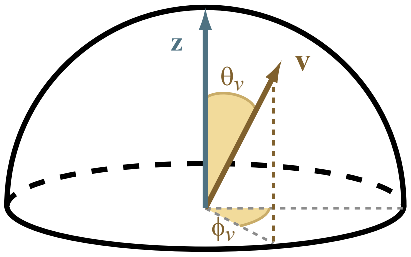


Figure 4: Illustration of the spherical coordinates $(\theta_v, \phi_v)$ for an arbitrary vector $\mathbf{v}$.

For continuous one-dimensional PDFs, we must map a uniform distribution of numbers to an infinite set of possible PDF proportional samples. This way, we can generate a random number from a uniform distribution and be more likely to obtain an important sample direction, just as more random values would map to the important sample S2. We can obtain the position of a sample $\Theta$ within the CDF by stacking all the previous samples before $\Theta$ on top of each other via integration,

$$P(\Theta) = \int_0^\Theta p(\theta) d\theta \tag{31}$$

To obtain the PDF-proportional sample from a random number, we set $P(\Theta)$ equal to a random value $\xi$ and solve for $\Theta$. In general, we denote the mapping from the random value to the sample direction distributed according to the PDF as $P^{-1}(\xi)$.

### 4.1.2 Phong BRDF Sampling

In the Phong BRDF, the glossy reflection is modeled by a cosine falloff centered around the specular reflection direction. We can represent the glossy component mathematically as $\cos^n \theta_s$, where $\theta_s$ is the angle between the sample direction and the specular direction and $n$ is the shininess of the surface,

$$L_o(\omega_o, \mathbf{x}) = \int_\Omega L_i(\omega_i, \mathbf{x}) \cos^n \theta_s \cos \theta_i d\omega_i. \tag{32}$$

To convert this material function into a PDF, we separate out the exponentiated cosine lobe from the illumination integral. The other components of the integrand are ignored for computational efficiency and mathematic simplicity. This results in a PDF that is in terms of the angles around the specular direction, $p(\theta_s, \phi_s)$. Here, $\theta_s$ and $\phi_s$ are the spherical coordinates of the sample direction in a coordinate frame where the specular direction is the z-axis (Figure 4). To formulate this PDF, we must first normalize the cosine lobe to integrate to one,

$$p(\theta_s, \phi_s) = \frac{\cos^n \theta_s \sin \theta_s}{\int_0^{2\pi} \int_0^{\pi/2} \cos^n \theta_s \sin \theta_s d\theta_s} = \frac{n+1}{2\pi} \cos^n \theta_s \sin \theta_s. \tag{33}$$

The extra sine appears when converting to spherical coordinates from solid angles. Often times, this normalization term is also included in the BRDF to ensure the reflectance model is energy conserving.

**Sample Directions.** To generate the two-dimensional sample direction $(\theta_s, \phi_s)$ according to this PDF, it is best to find each dimension of the sample direction separately [PH04b]. Therefore, we need a PDF for each dimension so we can apply Equation (31) to convert the PDF into a CDF and create a partial sample direction. To accomplish this, the marginal probability of the $\theta_s$ direction, $p(\theta_s)$, can be separated from $p(\theta_s, \phi_s)$ by integrating the PDF across the entire domain of $\phi_s$,

$$p(\theta_s) = \int_0^{2\pi} p(\theta_s, \phi_s) d\phi_s = (n+1)\cos^n\theta_s \sin\theta_s \tag{34}$$

This one-dimensional PDF can now be used to generate $\theta_s$. Given the value of $\theta_s$, we find the PDF for $\phi_s$ using the conditional probability $p(\phi_s|\theta_s)$ from Bayes' theorem [Bis06] defined as,

$$p(\phi_s|\theta_s) = \frac{p(\theta_s, \phi_s)}{p(\theta_s)} = \frac{1}{2\pi}. \tag{35}$$

The two probability functions are converted into CDFs by Equation (31) and inverted to map a pair of independent uniform random variables $(\xi_1, \xi_2)$ to a sample direction $(\theta_s, \phi_s)$,

$$\theta_s = \arccos\left(\xi_1^{\frac{1}{n+1}}\right), \tag{36}$$

$$\phi_s = 2\pi\xi_2. \tag{37}$$

Note that the integration for $p(\theta_s)$ is actually,$(1-\xi_1)^{1/(n+1)}$, but because $\xi_1$ is a uniformly distributed random variable between 0 and 1, $(1-\xi_1)$ is also uniformly distributed between 0 and 1. Moreover, this sampling strategy works for various other BRDFs such as the Lafortune reflectance model [LFTG97].

### 4.1.3 Exponential Distribution Sampling

Distributions driven by exponentials, including the Ward [War92] and Cook-Torrance [CT82] BRDF, cannot be analytically integrated like the Phong BRDF. Therefore, the inversion method will not work as presented in Section 4.1.2. As an alternative, we can use the Box-Muller transform [BM58] to sample the isotropic or anisotropic exponential distribution. The following provides an intuitive interpretation of the transform.

**Formulation.** In the two aforementioned BRDFs, the exponential distributions are based around the halfway vector $\mathbf{H}$ between the incoming $\omega_i$ and outgoing $\omega_o$ direction, where $\mathbf{H} = (\omega_i + \omega_o)/\,||\omega_i + \omega_o||$. The BRDFs are then defined to be approximately proportional to

$$f(\theta_h, \phi_h) \propto \exp\left(-\tan^2\theta_h\left(\frac{\cos^2\phi_h}{\alpha_x^2} + \frac{\sin^2\phi_h}{\alpha_y^2}\right)\right), \tag{38}$$

where $\theta_h$ and $\phi_h$ represent the zenith and azimuthal angle respectively (Figure 5) between the normal and the halfway vector and $\alpha_x$ and $\alpha_y$ are the roughness parameters, where $\alpha_x = \alpha_y$ when the reflection is isotropic.

**Sample Directions.** We start by sampling $\phi_h$ uniformly, similar to the Phong BRDF,

$$\phi_h = 2\pi\xi_2. \tag{39}$$

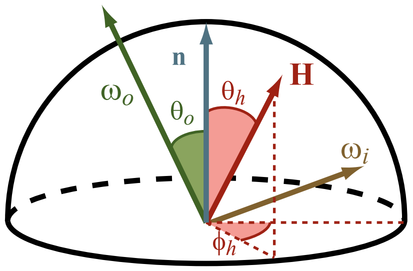


Figure 5: Illustration of the halfway vector and the corresponding variables used for the exponential distributions.

In the anisotropic case, this is insufficient since the distribution is elliptical instead of circular. So, we scale the directions accordingly using the tangent and arc tangent,

$$\phi_h = \arctan\left(\frac{\alpha_y}{\alpha_x}\tan(2\pi\xi_2)\right). \tag{40}$$

However, using the arc tangent requires special care for each quadrant and thus expensive conditionals in the shader logic. As an alternative, we can look at the definition of the tangent, and get $\cos\phi_h$ and $\sin\phi_h$ directly,

$$\begin{aligned}\tan\phi_h &= \frac{\alpha_y}{\alpha_x}\tan(2\pi\xi_2) && (41)\\ \frac{\sin\phi_h}{\cos\phi_h} &= \frac{\alpha_y\sin 2\pi\xi_2}{\alpha_x\cos 2\pi\xi_2} && (42)\\ \sin\phi_h &= \frac{\alpha_y\sin 2\pi\xi_2}{\sqrt{\alpha_x^2\cos^2 2\pi\xi_2+\alpha_y^2\sin^2 2\pi\xi_2}} && (43)\\ \cos\phi_h &= \frac{\alpha_x\cos 2\pi\xi_2}{\sqrt{\alpha_x^2\cos^2 2\pi\xi_2+\alpha_y^2\sin^2 2\pi\xi_2}}. && (44)\end{aligned}$$

This eliminates any unnecessary conditional logic in the shader. Please note that the denominator is required to normalize the scaled values and ensure the cosine and sine are on the unit circle.

Now that we have mapped the random value $\xi_2$ to $\phi_h$, we need to find a mapping of $\xi_1$ to $\theta_h$. Assuming we want to sample all values of the distribution, i.e. sample the range of $f(\theta_h,\phi_h)$, we can set the equation equal to the uniformly random variable $\xi_1$ and solve for $\theta_h$,

$$\begin{aligned}\xi_1 &= \exp\left(-\tan^2\theta_h\left(\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2}\right)\right) && (45)\\ -\log\xi_1 &= \tan^2\theta_h\left(\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2}\right) && (46)\\ \theta_h &= \arctan\sqrt{\frac{-\log\xi_1}{\frac{\cos^2\phi_h}{\alpha_x^2}+\frac{\sin^2\phi_h}{\alpha_y^2}}}. && (47)\end{aligned}$$

However, this only lets us know the position of the halfway vector. To determine the incoming light direction to use for sampling, we can convert $\theta_h$ and $\phi_h$ into a Cartesian coordinate representation of the halfway vector $\mathbf{H}$ and find $\omega_i$ using,

$$\omega_i = 2(\omega_o \cdot \mathbf{H})\mathbf{H} - \omega_o. \tag{48}$$

**PDF Value.** Given the sample direction $\omega_i$, we still must evaluate Equation (30) with a PDF value. As shown in [Wal05], this requires that the probability density of the two-dimensional Gaussian sampling must be warped into the space of the illumination integral. This warping operation can be done with using the Jacobian of the of the mapping.

The basic idea is that given two two-dimensional PDFs, $p_a$ and $p_b$, where the $i^{\text{th}}$dimension of $p_a$ can be mapped from $p_b$ using the function $a_i$ and the measure of probability from a region $\mathbf{B}$ in $p_b$ is equal to a the mapped region of $\mathbf{A_B}$ in $p_a$,

$$\int_{\mathbf{A_B}} p_a(a_1, a_2) da_1 da_2 = \int_{\mathbf{B}} p_b(b_1, b_2) db_1 db_2, \tag{49}$$

then the integrals can be related using the change of variables theorem from calculus. Here, the determinate of the Jacobian is used to convert the the variables,

$$\int_{\mathbf{A_B}} p_a(a_1, a_2) da_1 da_2 = \int_{\mathbf{B}} p_a(a_1(b_1, b_2), a_2(b_1, b_2)) \left| \frac{\partial a_1}{\partial b_1} \frac{\partial a_2}{\partial b_2} - \frac{\partial a_2}{\partial b_1} \frac{\partial a_1}{\partial b_2} \right| db_1 db_2. \tag{50}$$

In the case of the exponential distribution, we must first transform from the sample space to the halfway vector angles that are generated, giving

$$p(\mathbf{H}) = \left| \frac{\partial \xi_1}{\partial \theta_h} \frac{\partial \xi_2}{\partial \phi_h} - \frac{\partial \xi_2}{\partial \theta_h} \frac{\partial \xi_1}{\partial \phi_h} \right| \frac{1}{\sin \theta_h},$$

where the $\sin \theta$ compensates for the conversion between the spherical coordinate and solid angle space. This equation evaluates and simplifies to the the following PDF,

$$p(\mathbf{H}) = \left| \begin{array}{r} \left(\exp\left(-\tan^2 \theta_h \left(\frac{\cos^2 \phi_h}{\alpha_x^2} + \frac{\sin^2 \phi_h}{\alpha_y^2}\right)\right)\right) \left(\frac{-2 \sin \theta_h}{\cos^3 \theta_h} \left(\frac{\cos^2 \phi_h}{\alpha_x} + \frac{\sin^2 \phi_h}{\alpha_y}\right)\right) \\ \cdot \left(\frac{\alpha_x \alpha_y}{2\pi(\alpha_y^2 \cos^2 \phi_h + \alpha_x^2 \sin^2 \phi_h)}\right) \dfrac{1}{\sin \theta_h} - 0 \end{array} \right|,$$

$$p(\mathbf{H}) = \left| \left(\exp\left(-\tan^2 \theta_h \left(\frac{\cos^2 \phi_h}{\alpha_x^2} + \frac{\sin^2 \phi_h}{\alpha_y^2}\right)\right)\right) \left(\frac{1}{\pi \alpha_x \alpha_y \cos^3 \theta_h} \frac{\alpha_y \cos^2 \phi_h + \alpha_x \sin^2 \phi_h}{\alpha_y^2 \cos^2 \phi_h + \alpha_x^2 \sin^2 \phi_h}\right) \right|, \tag{51}$$

$$p(\mathbf{H}) = \frac{1}{\pi \alpha_x \alpha_y \cos^3 \theta_h} \exp\left(-\tan^2 \theta_h \left(\frac{\cos^2 \phi_h}{\alpha_x^2} + \frac{\sin^2 \phi_h}{\alpha_y^2}\right)\right).$$

Last, we multiply the Jacobian from the transformation from the halfway vector space to the sample direction space in Equation (48),

$$p(\omega_i) = \frac{p(\mathbf{H})}{4(\mathbf{H} \cdot \omega_i)}. \tag{52}$$

Combining Equation (51) and Equation (52), we obtain the final PDF for the anisotropic distribution for use in the illumination integral approximation,

$$p(\omega_i) = \frac{1}{4\pi \alpha_x \alpha_y \cos^3 \theta_h (\mathbf{H} \cdot \omega_i)} \exp\left(-\tan^2 \theta_h \left(\frac{\cos^2 \phi_h}{\alpha_x^2} + \frac{\sin^2 \phi_h}{\alpha_y^2}\right)\right). \tag{53}$$

## 4.2 Image-Based Environment Importance Sampling

As an alternative to BRDF sampling, we can also use an image-based environment to direct illumination integration. Various methods have been researched for efficiently integrating reflections from image-based environment light sources, such as the Voronoi cell-based approach in [ARBJ03], median-cut method presented in [Deb05], and the Penrose tiling-based approach in [ODJ04]. In the theme of this course and for code simplicity, we will provide an importance sampling treatment of image-based environment sampling, as presented by [PH04a]. Here, the sampling will be similar to the approach presented in Section 4.1.2 for the Phong BRDF sampling.

**Image to PDF Conversion.** The underlying idea in importance sampling the environment is to treat the image as a PDF. Sometimes this discrete version of a PDF is referred to as a *Probability Mass Function* (PMF). The PDF requires a mapping from the texture space of an image $(u, v)$ to the spherical coordinate space $(\theta, \phi)$. For programmatic simplicity, a latitude-longitude map provides a good choice where the texture coordinates are defined by linearly scaling the spherical coordinates,

$$\begin{aligned} \theta &= \frac{2\pi}{w}u \\ \phi &= \frac{\pi}{h}v. \end{aligned} \tag{54}$$

Here, $w$ and $h$ are the width and height in pixels of the texture space. Since we used an image where $u$ is equally spaced with respect to $v$, there is a large warping around the poles of the spherical map. This results from the radius diminishing around the azimuth angle at a rate of $\sin\theta$. When building the PDF, we compensate for this warping by attenuating the PDF by $\sin\theta$.

Environment images are often in color and the PDF is a scalar function of density. Thus, we need another mapping between the color of a texture element to a density value. A popular choice is to use the luminance value of the element, defined by,

$$L(u, v) = 0.2126R(u, v) + 0.7152G(u, v) + 0.0722B(u, v). \tag{55}$$

The luminance value represents a perceptually-based metric for how various color intensities are perceived by the human visual system [CIE31]. The basic idea behind using this mapping for the density is that by assigning a higher probability of sampling to the values with higher perceived intensity, the result will visually converge faster than using an arbitrary metric. Putting it all together, we have a PDF for the texture element $(u, v)$ defined as,

$$p(u, v) \propto L(u, v)\sin\theta. \tag{56}$$

Please note that the PDF is proportional and not equal to the right hand side since this currently represents an unnormalized function.

**Marginal and Conditional CDFs.** Given the PDF of the environment, we can now build the marginal and conditional CDFs as to sample the environment with two independent and uniformly distributed random variables $(\xi_1, \xi_2)$. The conditional CDFs are constructed by summing the PDF values for each column $u$,

$$P(v|u) \propto \sum_{i=1}^{v} p(u, i). \tag{57}$$

The marginal CDF is then computed by summing the maximum conditional values for each column $u$,

$$P(u) \propto \sum_{i=1}^{u} P(h|i). \tag{58}$$

Please note, this is not how conditional and marginal distributions are typically represented. Instead, these equations reflect how the CDF is computed in code. The CDFs are stored unnormalized to maximize numerical precision. This is addressed explicitly in the final sampling operation. Given a color image in latitude-longitude space, the following C++ code snippet produces the associated CDFs, given an image in the variable `m_pData` and the width, height, and the number of channels are defined respectively as `m_width` , `m_height`, and `m_channels`.

```
unsigned int k=0;
float angleFrac = M_PI / float(m_height);
float theta = angleFrac * 0.5f;
float sinTheta = 0.f;

float *pSinTheta = (float*) alloca(sizeof(float)*m_height);
for (unsigned int i=0; i < m_height; i++, theta += angleFrac)
{
    pSinTheta[i] = sin(theta);
}

// convert the data into a marginal CDF along the columns
float *pBuffer = malloc(m_width*(m_height+1)*sizeof(float));
float *pUDist = &pBuffer[m_width*m_height];
for (unsigned int i=0,m=0; i < m_width; i++, m+=m_height)
{
    float *pVDist = &pBuffer[m];

    k = i*m_channels;
    pVDist[0]  = 0.2126f * m_pData[k+0] + 0.7152f * m_pData[k+1] +
                 0.0722f * m_pData[k+2];
    pVDist[0] *= pSinTheta[0];

    for (unsigned int j=1,k=(m_width+i)*m_channels; j < m_height;
         j++, k+=m_width*m_channels)
    {
        float lum = 0.2126f * m_pData[k+0] + 0.7152f * m_pData[k+1] +
                    0.0722f * m_pData[k+2];
        lum *= pSinTheta[j];

        pVDist[j] = pVDist[j-1] + lum;
    }

    if (i == 0)
    {
        pUDist[i] = pVDist[m_height-1];
    }
    else
    {
        pUDist[i] = pUDist[i-1] + pVDist[m_height-1];
    }
}
```

Using the marginal and conditional CDFs defined by `pUDist` and `pVDists`, random variables are proportionally mapped to sample positions in the texture space $(u, v)$ using a binary search. The following snippet demonstrates how to sample the CDFs with the C++ STL binary search function `lower_bound`. Here, the random values are accessible in the vector `m_samples`.

```
float maxUVal = pUDist[m_width-1];

for (unsigned int  i=0; i < m_samples.size(); i++)
{
    float* pUPos = std::lower_bound(pUDist, pUDist+m_width,
                                    m_samples[i][0] * maxUVal);
    int u = pUPos - pUDist;

    float* pVDist = &pVDists[m_height*u];
    float* pVPos = std::lower_bound(pVDist, pVDist+m_height,
                                    m_samples[i][1] * pVDist[m_height-1]);
    int v = pVPos - pVDist;

    // store u,v into appropriate data structure
}
```

**Sample Space PDF.** Last, given the sample position in texture space, we must determine the PDF to evaluate the illumination integral in Equation (30). As in Section 4.1.3, we must convert from the texture space to the space of the illumination integral using the Jacobian of the mapping. Given the linear mapping in Equation (54), the Jacobian is simply,

$$\frac{w \cdot h}{2\pi}. \tag{59}$$

Since only the CDF is stored, the difference between the neighboring CDF values is used to compute the value of the PDF. Moreover, since the CDF is unnormalized, the PDF is divided by the last and largest value in the marginal and conditional CDF.

```
float angleFrac = M_PI / float(m_height);
float invPdfNorm = (2.f * float(M_PI*M_PI) ) / float(m_width*m_height);
float* pVDist = &pVDists[m_height*u];

// compute the actual PDF
pdfU = (u == 0)?(pUDist[0]):(pUDist[u]-pUDist[u-1]);
pdfU /= pUDist[m_width-1];

pdfV = (v == 0)?(pVDist[0]):(pVDist[v]-pVDist[v-1]);
pdfV /= pVDist[m_height-1];

float theta = angleFrac * 0.5 + angleFrac * y;
float invPdf = invPdfNorm / ( pdfU * pdfV ) * sin( theta );
```

### 4.3 Quasi-Random Low-Discrepancy Sequences

The problem with generating uniformly random sample directions using pseudo-random numbers is that the directions are not well distributed, leading to poor accuracy of the estimator in Equation (30). We can improve the accuracy by replacing pseudo-random numbers with quasi-random low-discrepancy sequences that intrinsically guarantee well-distributed directions [Kel01]. One such sequence is known as the *Hammersley* sequence. A random pair $(\xi_1, \xi_2)$ in the two-dimensional Hammersley sequence with $N$ values is defined as,

$$(\xi_1, \xi_2) = \left(\frac{i}{N}, \Phi_2(i)\right), \tag{60}$$

where the radical inverse function, $\Phi_2(i)$ returns the number obtained by reflecting the digits of the binary representation of $i$ around the decimal point as illustrated in the following table.

| | **Binary** | **Inverse Binary** | **Value** |
|---|---|---|---|
| 1 | 1 | 0.1 | 0.5 |
| 2 | 10 | 0.01 | 0.25 |
| 3 | 11 | 0.11 | 0.75 |
| 4 | 100 | 0.001 | 0.125 |

As provided in [KK02], this can be coded by a simple bit swapping operator in C.

```
double HammersleyRadicalInverse(unsigned int bits)
{
    bits = ( bits << 16 ) | (bits >> 16);
    bits = ((bits & 0x00FF00FF) << 8 ) | ((bits & 0xFF00FF00) >> 8 );
    bits = ((bits & 0x0F0F0F0F) << 4 ) | ((bits & 0xF0F0F0F0) >> 4 );
    bits = ((bits & 0x33333333) << 2 ) | ((bits & 0xCCCCCCCC) >> 2 );
    bits = ((bits & 0x55555555) << 1 ) | ((bits & 0xAAAAAAAA) >> 1 );

    return (double) bits / (double) 0x100000000LL;
}
```

Another advantage of using low-discrepancy sequences is that we can reuse them for each pixel. The deterministic sequence ensures that no matter how many times the frame is rendered, the result will always be the same. Moreover, this makes the algorithm amenable for GPU rendering since we can pre-compute the low-discrepancy sequence and transfer them to the GPU using a constant buffer. Unfortunately, the deterministic sample sequences will turn noise into replicated specular reflections, or *alias* (Figure 6(a)). However, the aliasing can be suppressed using filtered importance sampling as explained in Section 5.

## 5 BRDF Proportional Filtered Importance Sampling

While importance sampling and low discrepancy sequences reduce the variance of the illumination integral, a large number of samples are still required to produce a smooth solution. If using a deterministic sampling sequence, the noise will appear as aliasing when performing BRDF proportional sampling. To remove the alias in the signal, we borrow from a standard method in signal processing and pre-filter the incoming signal (Figure 6(c)). However, to determine the support, or size, of the filter we need a computationally efficient function capable of estimating the filtered region without over-filtering and dulling the reflectance or under-filtering and leaving the alias.

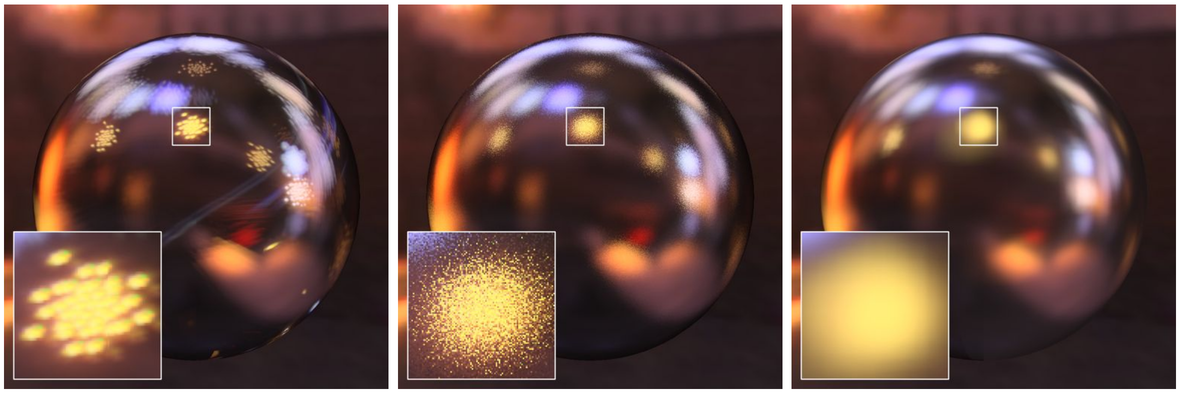

(a) (b) (c)

Figure 6: Deterministic importance sampling for every pixel causes an aliasing effect in the illumination integral estimate (a). Typically, random samples are used for each pixel providing a noisier, but more visually acceptable result (b). With filtered importance sampling, we can use computationally efficient, deterministic sampling while obtaining smooth results (c).

## 5.1 Foundation

If the PDF of a sample direction is small (that is to say, if the sample is not very probable), then it is quite unlikely that other samples will be generated in a similar direction. In such a case, we want the sample to bring illumination from the environment filtered over a large area, thereby providing a better approximation of the overall integration. On the other hand, if the PDF of a direction is high, other samples are likely to be generated in similar directions, thus multiple samples will help average out the error in the integration estimation from this region. In this case, the sample should only average a small area of the environment.

We define this relationship in terms of the solid angle associated with a sample $\Omega_s$, computed as the inverse of the product between the PDF and the total number of samples, $N$,

$$\Omega_s = \frac{1}{N \cdot p(\omega_i, \omega_o)}. \tag{61}$$

However, this only provides a scalar solid angle and not the shape of the filter region. As an approximation, we assume that the filter region is circular around the incoming light direction $\omega_i$, i.e. the filter is *isotropic*.

This approximation may cause some samples to gather light from inappropriate regions and thus over-filter the sample. However, the isotropy is simple and fast to compute as seen in the following sections. Moreover, any over-filtering can be easily addressed by adding more samples. For a theoretical foundation for this relationship and filter approximation, we refer the reader to [KC08].

Given Equation (61) and the BRDF importance sampling strategies discussed in Section 4.1, we must determine how to filter the various light sources. Effectively, we must determine a function that maps $\Omega_s$ to an area on a light source that can be efficiently filtered. The following sections define this mapping for two illumination models, image-based environment and area light sources.

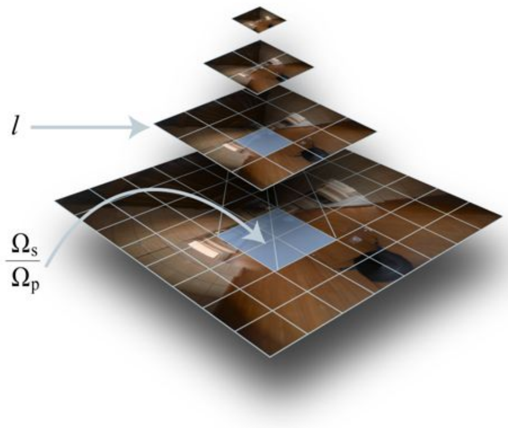


Figure 7: A visualization of a MIP-mapped environment texture, where $\Omega_s/\Omega_p$ is shown on the $0^{\text{th}}$ level of the environment map and the corresponding filtered sample is highlighted on level $l$.

## 5.2 Environment Lights

**MIP-Map Filtering.** When pre-filtering with environment lights, we should filter all pixels of the map within the solid angle $\Omega_s$ from Equation (61) around the sample direction. To make the filtering efficient, we use the MIP-map data structure [Wil83] commonly used for anti-aliasing textures. A MIP-map is a pyramidal set of images, where each pixel of level $l$ is the filtered result of the four corresponding pixels of level $l-1$ (Figure 7) and level zero is the original texture. Therefore, filtering all the pixels within the solid angle $\Omega_s$ can be approximated by choosing an appropriate level of the MIP-map.

First, we compute the number of environment map pixels in $\Omega_s$. Assuming the shape is isotropic (or circular), this is given by the ratio of $\Omega_s$ to the solid angle subtended by one pixel of the $0^{\text{th}}$ MIP-map level, $\Omega_p$. This, in turn, is given by the texture resolution, $w \cdot h$, and the mapping distortion factor $d(\omega_i)$,

$$\Omega_p = \frac{d(\omega_i)}{w \cdot h}. \tag{62}$$

The distortion factor corresponds to the size of a mapping in a certain direction $\omega_i$. This term compensates for the stretching at the poles in a latitude-longitude map or the outer edges of a dual paraboloid map. However, the distortion may be complex to derive and compute. In practice, we found that equally weighting each direction produces similar results to those obtained using correct distortion.

As aforementioned, we want the environment map lookup to correspond to averaging $\Omega_s/\Omega_p$ pixels, which yields the formula for the MIP-map level,

$$l = \max\left[\frac{1}{2}\log_2\frac{\Omega_s}{\Omega_p}, 0\right] = \max\left[\frac{1}{2}\log_2\left(\frac{w \cdot h}{N}\right) - \frac{1}{2}\log_2\left(p(\omega_i,\omega_o)d(\omega_i)\right), 0\right]. \tag{63}$$

For each sample direction, the filtering operation is reduced to evaluating Equation (63), and using $l$ in the corresponding texture call. In NVIDIA's GPU language Cg, $l$ is fed to the function `tex2Dlod()` as the level of detail parameter when looking-up the environment map.

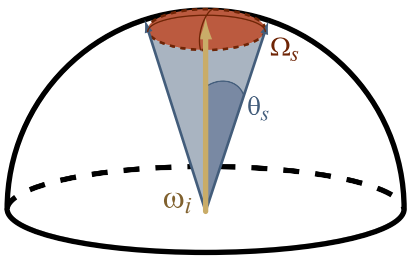


Figure 8: The parameters of the spherical cap used as the isotropic filter region for area lights.

**Ptex Filtering of Cube Maps for RenderMan.** In production renderers, such as Pixar's RenderMan, other intrinsic data structures can be used to achieve better filtering with less distortion than the proposed MIP-map filtering method. One option is the Per-face Texture (Ptex) environment representation [BL08]. Here, the Ptex format will store the environment as a cube map. This mapping provides relatively minimal distortion since the environment is projected onto each face of the cube, which is similar to GPU cube maps. However, unlike GPU cube maps, the Ptex algorithm will filter across the various faces of the cube map. This will provide a smooth, seamless transition when filtering a region present on multiple faces of the cube.

In Pixar's RenderMan, the filtered cube maps are accessed using the `environment()` function call. Here, the environment is passed four three-dimensional vectors representing the filter region. However, generating the filter region cannot be analytically computed as with the MIP-map level in the GPU approach.

Instead, we generate a spherical cap (Figure 8), and orient the filter vectors around the cap. This once again represents our isotropic filter approximation. Here, we assume the filter region lies within a spherical cap with an area of $\Omega_s$ and the entire cap is above the horizon. The bounds of the spherical cap with respect to the angle between the sample vector, $\theta_s$, can be found by solving the equation for the area of a spherical cap, $\Omega_s = 2\pi(1 - \cos\theta_s)$, for $\theta_s$ giving,

$$\theta_s = \arccos\left(1 - \frac{\Omega_s}{2\pi}\right). \tag{64}$$

The four vectors representing the filter region are then computed by rotating the sample direction by $\theta_s$ around two orthogonal directions (Figure 9). Given two orthogonal vectors, `tV` and `bV`, and the sample direction `wi` the filter region can be computed using the following RSL code snippet.

```
float cosThetaCap = 1.0-1.0/(pdf * numSamples * 2.0 * PI);
float sinThetaCap = sqrt( 1.0 - cosThetaCap*cosThetaCap );

vector cosThetaCapWi = cosThetaCap * wi;

vector wi_tp =  cosThetaCapWi + sinThetaCap * tV;
vector wi_tn =  cosThetaCapWi - sinThetaCap * tV;
vector wi_bp =  cosThetaCapWi + sinThetaCap * bV;
vector wi_bn =  cosThetaCapWi - sinThetaCap * bV;

color filteredValue = environment( ptexEnvironmentMap, wi_tp, wi_bp,
                                   wi_tn, wi_bn,
```

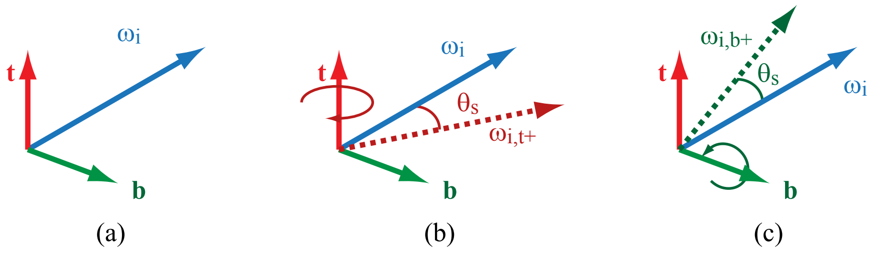


Figure 9: Illustration of the rotation of $\omega_i$ around the orthogonal vectors $\mathbf{t}$ and $\mathbf{b}$ to generate the filter region vectors or the ray differentials.

```
                    "filter", "gaussian" );
```

Please note that the filter is forced to be Gaussian to prevent any sharpening artifacts from occurring by possible negative weight filters being set as the default in the renderer. Moreover, orthogonal vectors `tV`, `bV` and `wi`, i.e. the coordinate frame, must be smooth across the image. Since filtering is effectively adding bias to the simulation, any large discontinuities in the coordinate frame will result in visual artifacts.

**Smooth Coordinate Frame.** When using the MIP-map based filtering, the coordinate frame is implicitly defined to be oriented in the texture space of the environment. When specifying the coordinate frame from an arbitrary sample vector, this requires more care to ensure the orientation vectors are continuous and smooth across the rendered image. To find these orthogonal vectors, denoted as $\mathbf{t}$ and $\mathbf{b}$, we start with the guess vector $\mathbf{b}_{\text{guess}}$. Given this guess vector, we build a coordinate system using the orthogonality of the cross product operator,

$$\begin{aligned}\mathbf{t} &= \mathbf{b}_{\text{guess}} \times \omega_i \\ \mathbf{b} &= \omega_i \times \mathbf{t}.\end{aligned}$$

As $\mathbf{b}_{\text{guess}}$ becomes parallel to $\omega_i$, a pole-like discontinuity artifact appears in the filtered reflectance. A common solution is to use another guess vector that is orthogonal to the degenerate case. This works well for stochastic sampling because the noise covers the flip. When using a large number of samples, the results will converge on the exact solution which will also mask the flip. Unfortunately, deterministic FIS introduces a bias and this flip appears as a tear in the reflection. Since $\partial P \partial u$ and $\partial P \partial v$ contain discontinuities between faces, the partial derivatives available in production renderers such as Pixar's RenderMan are also ineligible to be the guess vector. Similarly, texture coordinate based differentials may have discontinuities depending on the layout of the UVs. The key here is to recognize the discontinuity always exists due to the underdetermined nature of the single vector coordinate system problem. Therefore, the goal is to hide the degenerate case from the camera.

One approach is to anchor the coordinate system to the camera-visible geometry. This way, the surface curvature masks any artifacts due to a changing guess vector. As an indicator of this curvature, we can use the camera space normals $\mathbf{n}$.

However, the normals themselves do not make a good guess vector. Using $\mathbf{n}$ may result in a degenerate case when $\omega_i$ is parallel to $\mathbf{n}$. Alternatively, we choose the partial derivative of $\mathbf{n}$ with respect to the spherical direction $\phi$. If we convert $\mathbf{n}$ to spherical coordinates such that $\mathbf{n}$ is equal to $(\cos\phi\sin\theta, \cos\theta, \sin\phi\sin\theta)$ and find the vector with

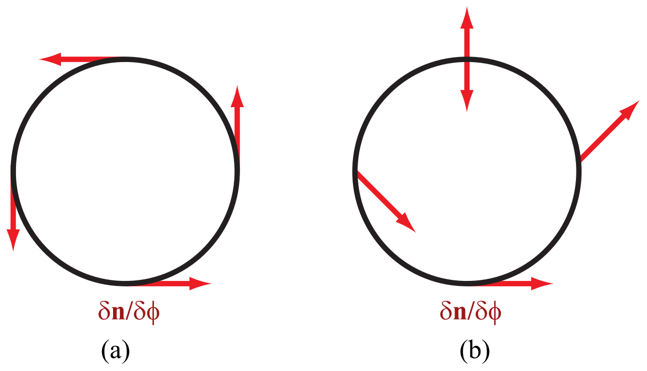


Figure 10: The partial derivative of the normal in camera space, $\partial\mathbf{n}/\partial\phi$, is taken (a) and stretched (b) to produce a guess vector for building coordinate frames that is smooth and anchored to the geometry and viewing direction.

respect to $\partial\phi$, we obtain the guess vector, $(-\sin\phi\sin\theta, 0, \cos\phi\sin\theta)$. When using this guess vector with the reflection vector, a discontinuity commonly happens around the silhouette edge of the object. In order to eliminate the discontinuity, we stretch the guess directions such that the discontinuity occurs behind the silhouette edge. This is done by simply scaling and offsetting $\theta$ and $\phi$ via $\theta' = \theta/2+\pi/4$ and $\phi' = 2(\phi-\pi/2)+\pi/2$. Using this method, we obtain a geometry-anchored, temporally smooth coordinate frame for isotropic surface reflection. Please note that this method would not work for anisotropic surfaces since the coordinate frame depends on the viewing direction.

## 5.3 Area Lights

Area lights represent a higher dimensional mapping problem than the environment lighting model. In this case, for each shading point, both the position and orientation of the light determine the filter region. Instead of trying to find an analytical mapping as with the environment lights, we opt for an approximate numerical solution similar to the computed filter region used for image-based environments.

**Mapping.** The goal here is to find the region on the light that corresponds to $\Omega_s$ of the BRDF sample from Equation (61). This can accurately be done by projecting the shape of the sample region onto the light's plane. However, this projected shape can be computationally expensive to evaluate. As an approximation, we assume the filter is isotropic. With respect to the hemisphere of incoming light directions, the isotropic filter is a spherical cap centered in the sample direction and is entirely above the horizon (Figure 8).

**Ray Differentials.** As an algorithmically simple approach for projecting the spherical cap, we borrow from the texture filtering methods in ray tracing and use ray differentials [Ige99]. Here, we send two rays alongside the sample ray to determine the filter region on the area light. These supplemental rays are called *ray differentials*. The ray differentials are rotated by $\theta_s$ from the sample's orientation in a direction orthogonal to the original sample (Figure 9). In this case, the sample direction is the incoming light direction $\omega_i$ and the ray differentials are $\omega_i$ rotated by $\theta_s$ in the direction of the smooth orthogonal vectors discussed in Section 5.2. We shoot the ray differentials and obtain

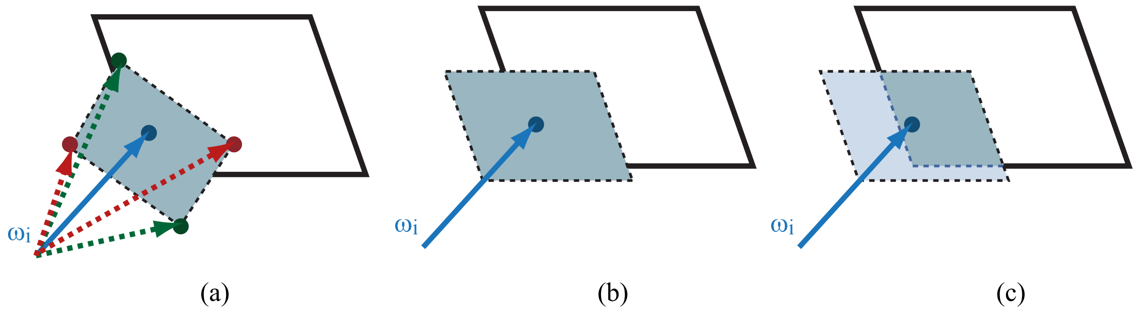


Figure 11: In determining the filter region of an area light sample, the sample ray and ray differentials intersect the plane of the area light (a). The intersecting area is used to build an isotropic filter region, i.e. an area light aligned box (b). The region is clipped by the bounds of the area light and used for filtering (c).

a rectangle in the plane of the light which represents the filter region. To provide an isotropic filter response, the area of the filter region is computed and used to generate an area light-aligned box (Figure 11).

**Filtering.** The simple box filter can be computed by taking the ratio of the area of the clipped filter region over the area of the light source. The clipping can become moderately complex to compute if the rectangle is not aligned with the area light coordinate system. To minimize the computation, the rectangle from the ray differentials is axis aligned thereby reducing the clipping operation to a series of min and max operations. However, this low quality filter produces noticeable artifacts that stem from the approximations made to compute the filter region. As a better filter and estimate of the projected elliptical shape of $\Omega_s$, we use a Gaussian filter kernel.

Due to the axis aligned approximations made for fast clipping, we perform a relatively cheap integration of the filter region with the Gaussian kernel. For instance, if the filter region is translated such that the center of the clipped rectangle is at zero and if the clipped region is scaled by the width and height of the entire filter region, the problem can be formulated as,

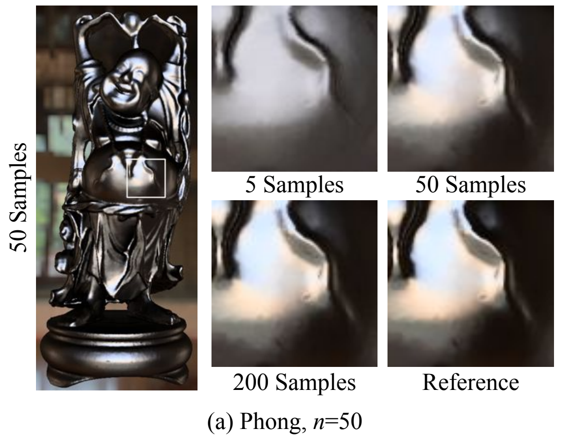
$$\frac{1}{2\pi}\int_{-x}^{+x}\int_{-y}^{+y}\exp(-x^2-y^2)dxdy, \tag{65}$$

where $-x, +x, -y$, and $+y$ represent the scaled bounds of the clipped filter region. The function then has an analytical integral in terms of the error function `erf` ,

$$\frac{1}{4}(\text{erf}(-x)-\text{erf}(+x))\cdot(\text{erf}(-y)-\text{erf}(+y)). \tag{66}$$

Thus, the filtering convolution is reduced to four `erf` function calls, which has approximately the same computational complexity as four `exp` calls.

**Slide Maps.** Artists may also want to put a slide map, or gobo, in front of the light source to add texture to the light color. Since the filter area with respect to the entire area light is known, the number of pixels is computed by scaling the ratio between the filter area and the area light size by the resolution of the slide map. The number of pixels are then converted into a MIP-map level using a formula similar to Equation (63) and the filtered texture value is accessed using the same method as described in Section 5.2.



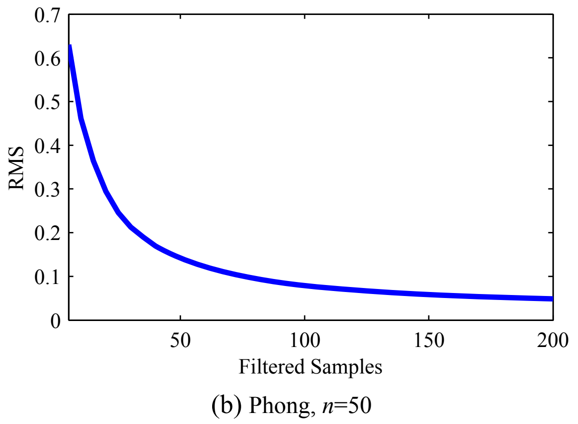


Figure 12: On complex geometry, undersampling appears as blurring and dulling of the highlights although appearing visually acceptable around 50 samples (a). The blurring on complex geometry affects the RMS (b) such that more samples are required to obtain an error similar to the spheres in Figure 14. However, due to the masking effect of the high-frequency contours, the error is less noticeable than on smooth surfaces.

## 5.4 Analysis

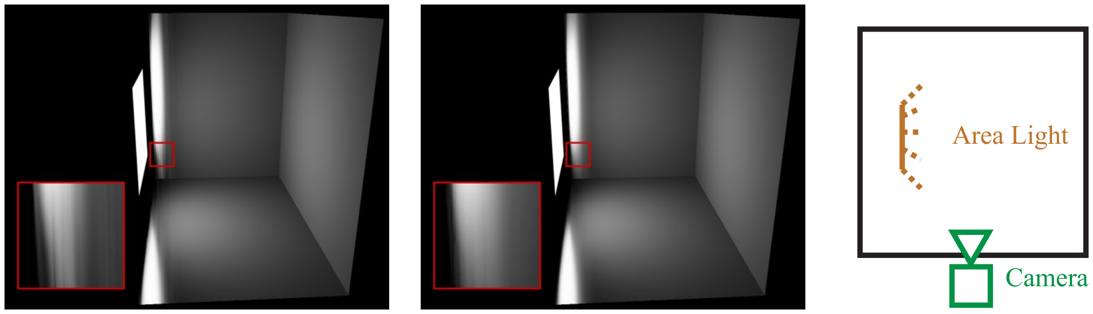
Using the aforementioned methods, we can obtain accurate smooth glossy surface reflections with a relatively small number of samples. The following analyzes the resulting artifacts when using too few samples with area and image-based environment light sources. We discuss the convergence pattern associated with smooth and complex geometry on a few anecdotal cases.

**Environment Lights.** When viewing BRDF-based FIS integration with the environment light, undersampling causes a glossy material to appear duller (Figure 12). This is due to filter kernels not faithfully respecting the shape of the BRDF. This can cause seams to appear on smooth surfaces when using the GPU-based implementation. The seams occur from the image representation of the environment which has edge boundaries. When using with latitude-longitude environment parameterizations, the edge occurs along the left- and right-most regions of the maps. For the dual paraboloid parameterization, the artifacts occur when filtering in the transition region between the two hemispheres. In both cases, the seam artifacts appear since the reconstruction filter is performed in image space and not in a spherical environment space.

[CK07] suggest using a scaled dual paraboloid environment parameterization to reduce this artifact. The scale results in a portion of the alternate side of each paraboloid to appear in the other's map, which affords limited filtering across hemisphere boundaries. However, this still produces a seam artifact if the scale is too small or when using too few samples.

As mentioned in Section 5.2, the Ptex-based environment representation removes these artifacts by filtering across the boundary regions. In practice, if the reflector is rough due to displacement or a bump map, the latitude-longitude mapping can be sufficient since the seam will be masked by the surface variation.

**Area Lights.** Similar to environment lights, area lights also have an overall dulling when using too few samples. However, area lights do not have a seam associated with a



(a) 16 samples (b) 128 samples (c) Top view of scene

Figure 13: Area light aliasing artifact. Here, the undersampling appears as ripples in the reflection (a) and can be reduced by sending more samples from the BRDF (b).

filter boundary. Instead, the glossy area light reflections alias when only a small number of samples hit the emitter. This is evident when viewing reflecting surfaces that are perpendicular to the area light (Figure 13). The result is a low frequency, shimmering artifact that occurs until enough samples ray hit the light source. However, due to the low frequency property, the artifact tends to be masked by high frequency variations in the surface normal.

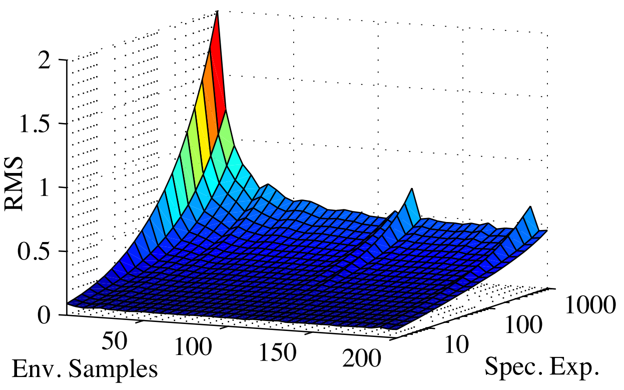
**Convergence Rates.** While these artifacts appear when using too few samples, visually acceptable results typically appear around 40 to 60 samples. In Figure 14, the root mean squared (RMS) error is compared between BRDF-based filtered importance and regular environment map sampling. One measure of success for the method is to analyze the error with respect to the number of samples. This is called the *convergence rate*. As illustrated, FIS works well with glossy surfaces and respects the $N^{1/2}$ convergence rate associated with Monte Carlo quadrature. However, unlike standard Monte Carlo quadrature, the results are significantly smoother and require fewer samples for a visually pleasing solution.

When comparing the results to deterministic environment map sampling, BRDF-based FIS performs better with glossy surfaces whereas environment sampling produces better matte reflections with a small number of samples. In practice, as the BRDF becomes more diffuse, the effectiveness of BRDF-based FIS wanes as the artifacts become more prevalent.

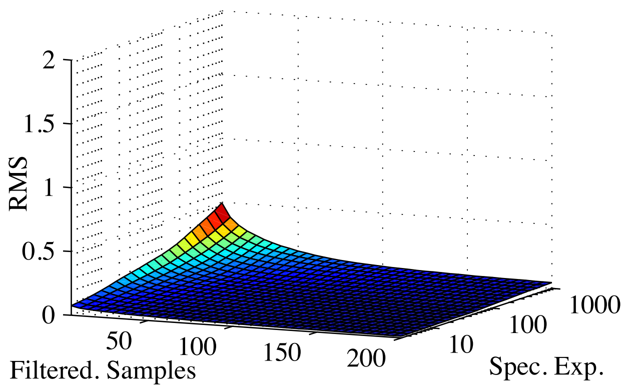
When looking at more complex geometry, such as the Buddha statue in Figure 12, we see a practical use case for FIS, where the only noticeable artifact is the dulling of the highlights. The high frequency contours of the surface cover up any seam artifacts and the results look visual pleasing around 50 samples per pixel.

This method extends well for limited amount of anisotropic reflection, as seen in Figure 15. Unfortunately, as the amount of anisotropy increases the isotropic filter approximations become insufficient. The result is the appearance of replicated specular reflections along the direction of high anisotropy. As an alternative, one can use anisotropic texture filtering. This is typically done by specifying the region of the filter instead of just controlling a blur or MIP-map parameter. However, this often does not provide good results.

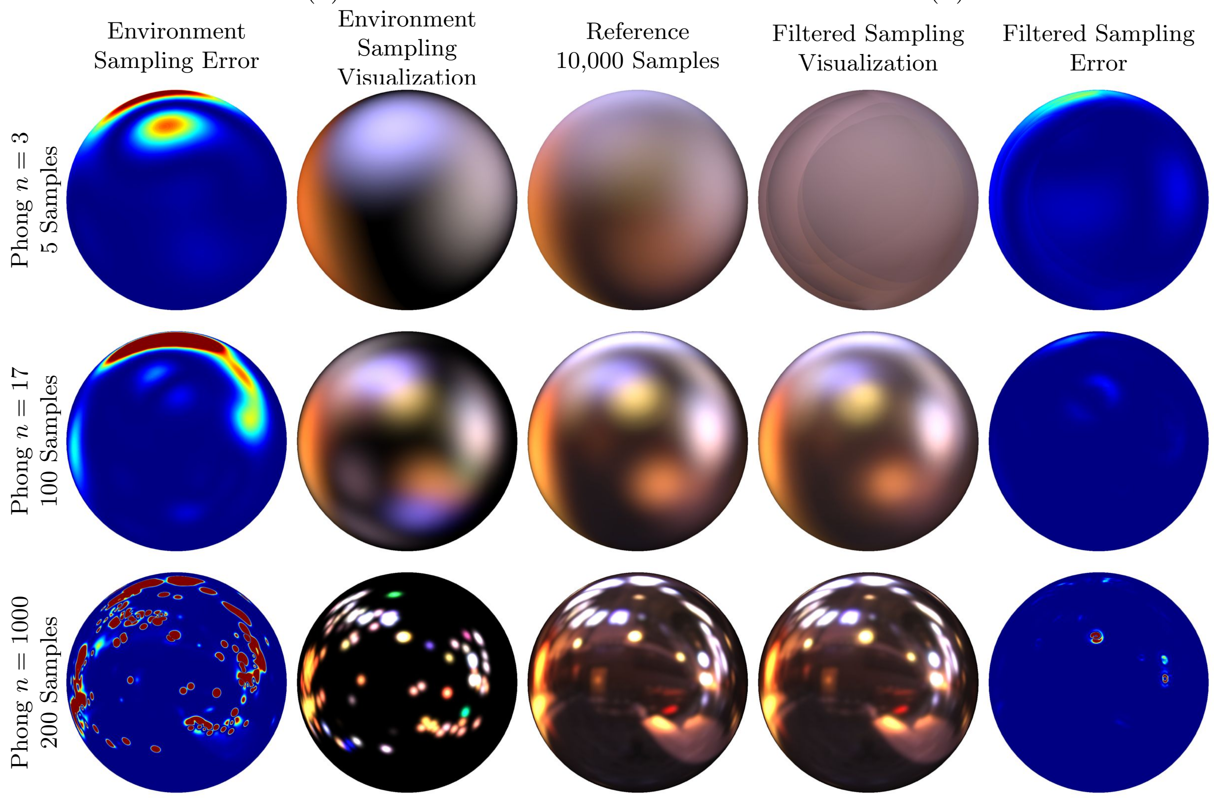
In practice, the sampling requires a small amount of over-filtering, or overlap on the filter region between samples. This property is intrinsically afforded by the isotropic approximations. Care must be taken in order to perform this operation with an anisotropic



(a)



(b)


Environment Sampling Error
Environment Sampling Visualization
Reference 10,000 Samples
Filtered Sampling Visualization
Filtered Sampling Error
Phong $n = 3$ 5 Samples
Phong $n = 17$ 100 Samples
Phong $n = 1000$ 200 Samples

(c)

Figure 14: Comparison of filtered BRDF and environment importance sampling. Panel (a) shows the RMS error when rendering a sphere lit by the grace light probe. While both methods converge at a rate of $N^{-1/2}$, filtered BRDF importance sampling has a lower RMS overall, especially on more glossy materials. Visually, the BRDF importance sampling better captures many of the subtle features of the environment (b). Although the RMS figures suggest better performance for BRDF importance sampling, even for low-frequency glossy reflections with few samples, environment IS provides more visually pleasing images in such cases (c).

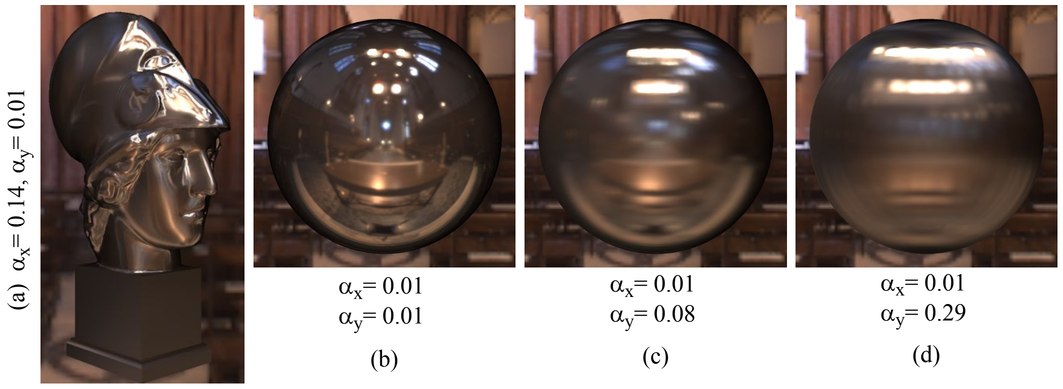


Figure 15: The Ward anisotropic BRDF rendered using 40 filtered importance samples. The method works well on complex surfaces (a) as well as high frequency materials (b) with slightly anisotropic reflections (c). However, it breaks down when rendering more anisotropic materials (d) due to our approximate isotropic filter.

filter and often the cost of anisotropic filtering outweighs the visual benefit. Instead, a few more isotropic samples will typically capture the reflection.

## 6 Multiple Importance Sampling

Recall that one of the main objectives in rendering is to approximate the illumination integral in Equation (29):

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega L_i(\mathbf{x}, \omega_i) f(\omega_o, \omega_i) \cos\theta_i d\omega_i$$

which can be split into three components: incoming illumination $L_i$, cosine weighted BRDF $B$ and visibility function $V$. If we drop the spatial and angular, the illumination integral becomes

$$L_o = \int_\Omega L_i B V d\omega_i \tag{67}$$

and the traditional Monte Carlo estimator is

$$\hat{L}_o = \frac{1}{N} \sum_{i=1}^{N} \frac{L_i B V}{p(\omega_i)} \tag{68}$$

where $p(\omega_i)$ is the importance sampling density. Ideally, the density function would be proportional to the product of $LBV$. Unfortunately, this is impossible for all but some artificially contrived scenes. We have to resort to some other density that will hopefully generate low variance in estimate. Let us examine a few possible options.

When we have a diffuse BRDF and multiple area lights of different sizes, we have two obvious choices for importance sampling densities. We can either sample according to the diffuse BRDF or lighting. Figure 16 illustrates the two scenarios. Using the diffuse BRDF, we sample the entire hemisphere, but only small portions of the hemisphere contain any lighting. So, many samples are completely wasted since the contribution will be zero. On the other hand, if we sample according to the lighting, none of the samples will be wasted because for any direction in which light is emitted the BRDF

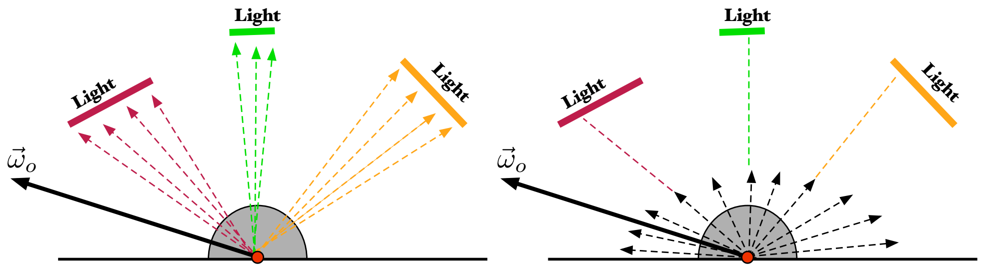


Figure 16: Diffuse BRDF and area lights. When we have a diffuse BRDF and multiple area lights of different sizes, two obvious sampling density choices are lighting (left) and BRDF (right). Note that BRDF sampling produces many samples that will be wasted, because there is no light emission in those directions. Sampling according to only the lighting produces lower variance.

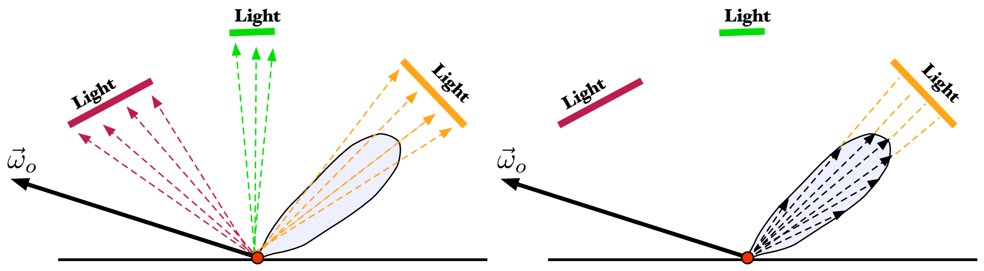


Figure 17: Glossy (specular) BRDF and area lights. Lighting (left) or BRDF (right) can be used as an importance sampling density. Note that light sampling produces many wasted samples because there will be no reflection in those directions. Sampling according to the BRDF provides better results.

will reflect some light. For diffuse surfaces, it is better to sample according to lighting only.

When we have glossy surfaces and many area lights, we can also sample according to the glossy BRDF or lighting densities. Figure 17 shows two sampling scenarios. In contrast to diffuse surfaces, glossy surfaces reflect light from a small solid angle. Using light sampling densities, most of the samples will be wasted because the surface will not reflect any light from those directions. Therefore, a better choice is to sample according to the glossy surface reflection, because there is a much larger chance that at least some sampled directions within a reflectance cone will have non-zero lighting contributions.

When we have very glossy surfaces or diffuse only surfaces, the choice of sampling densities is fairly obvious. As highly glossy surfaces become duller (more diffuse) the choice becomes murkier and not straightforward. For slightly glossy surfaces, a combination of two sampling strategies should be used.

So far, we have only looked at idealized situations where we have simple surfaces (composed of simple BRDFs) and no occluders. This is obviously an unrealistic situation. As shown in Figure 18 for a diffuse only surface, once we add occluders to the scene the sampling strategy becomes more complicated. Occluders can prevent light reaching the surface. Sampling according to lighting will generate proper directions, but lighting from those directions might be blocked. For the time being, we ignore visibility in our sampling density but we will return to it later and discuss what we could do to incorporate visibility into the sampling density.

It is clear that complex lighting, surface properties, and occlusions cause the function

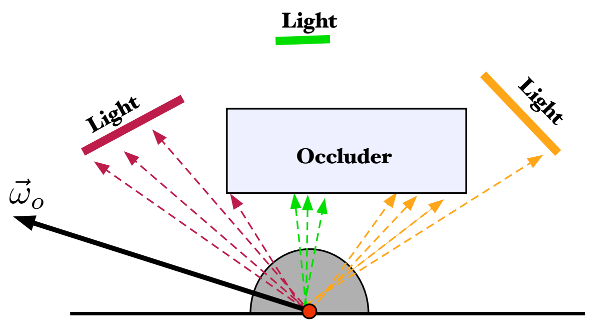


Figure 18: Diffuse BRDF and area lights with occluder. While sampling according to lighting is still preferable, the occluder blocks many of the directions that contribute light. Ideally, we would not pick directions that are blocked by the occluder. Unfortunately, this cannot be easily achieved for arbitrary scenes and geometry.

we want to approximate to be complex and discontinuous. This function can have many bright and dark regions and intensities can differ by orders of magnitude. Since the function is very complex and does not have a nice formulation (due to occlusion) it is clear that either our sampling density will be complex or that we need more than one sampling density.

Veach and Guibas [VG95] have demonstrated that by combining multiple sampling strategies, the variance can be reduced in situations where a single sampling strategy is bad (see Figure 19).

**How do we implement Multiple Importance Sampling?** Given the two sampling strategies for lighting and BRDF discussed in Section 4, let $p_1(x)$ and $p_2(x)$ be a BRDF and light sampling density. The random variables $X$ and $Y$ are then

$$X_{1,i} \sim p_1(x) \qquad X_{2,i} \sim p_2(x)$$
$$Y_{1,i} = \frac{f(X_{1,i})}{p_1(X_{1,i})} \qquad Y_{2,i} = \tfrac{f(X_{2,i})}{p_2(X_{2,i})}.$$

Now, we just need to combine the samples together:

$$Y_i = w_1 Y_{1,i} + w_2 Y_{2,i}. \tag{69}$$

The only remaining question is how to compute weights $w_i(x)$. We have already mentioned a few possible options in Section 2. One is using the balance heuristic, where the weights are:

$$w_i(x) = \frac{p_i(x)}{p_1(x) + p_2(x)} \tag{70}$$

and the final PDF $p(x)$ for the combined sampling densities is:

$$p(x) = w_1(x)p_1(x) + w_2(x)p_2(x). \tag{71}$$

Now, we have all the ingredients to implement multiple importance sampling.

One of the remaining questions is how do we choose the number of samples for each sampling strategy. There are several possibilities:

- Select a fixed number of samples for each strategy. For example, if $N = 100$, then $N_1 = 50$ would be used for lighting sampling and $N_2 = 50$ for BRDF sampling. Note that this is a relatively safe choice, although it could lead to suboptimal sample generation. If the BRDF is very glossy, some of the samples might be wasted, because too many samples are allocated for lighting sampling.

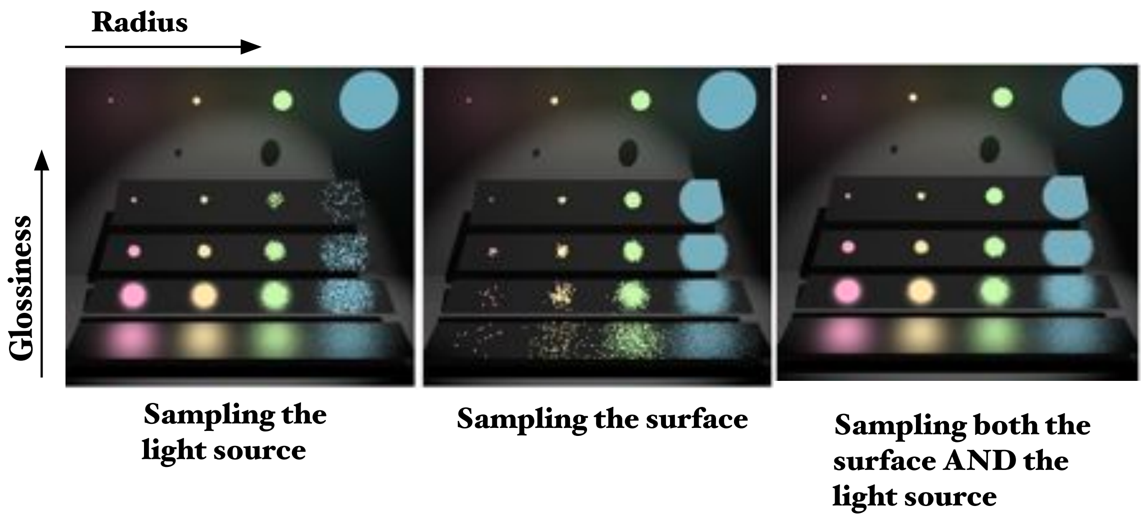

Radius
Glossiness
Sampling the light source
Sampling the surface
Sampling both the surface AND the light source

Figure 19: Combining many sampling strategies using Multiple Importance Sampling (MIS) produces superior results to using a single sampling density. Image from Veach and Guibas [VG95].

- Alternatively, the number of samples for each sampling strategy can be adjusted based on a heuristic, such as a combination of the solid angle of the light and glossiness of the surface. A reasonable strategy might be to have a minimum number of samples that will be taken according to each strategy and then distribute the rest based on the heuristic. For instance, if $N = 100$, we might allocate 20 samples to each sampling strategy. The remaining 60 samples would be distributed based on the glossiness of the surface and the light's solid angle.

**Notes.** Multiple importance sampling is an unbiased method for reducing the variance of Monte Carlo estimators. However, if it is used in conjunction with filtered importance sampling (*e.g.*, filtered importance sampling is used to filter environment lighting) the method is *biased* due to the nature of FIS. For visual effects applications, this is not troublesome as the noise can be greatly reduced.

While MIS reduces the variance, there are still configurations where the variance will be high. As Kollig and Keller [KK00] pointed out, multiple importance sampling attempts to hide a weakness of using a single density function. If, however, only one sampling density exists for some region of our integration domain $\Omega$, the multiple importance sampling will revert to a standard importance sampling. Kollig and Keller call this an *insufficient set of techniques* [KK00]. We emphasize again, if inappropriate sampling densities are chosen, multiple importance sampling will not help to reduce the variance.

## 7 Practical Notes on Monte Carlo Sampling

### 7.1 Choosing Sampling Density

The effectiveness of importance sampling depends on the choice of the importance sampling density $p(x)$. Figure 20 shows the differences between uniform and importance sampling.

Figure 21 illustrates why the choice of importance sampling density is crucial for

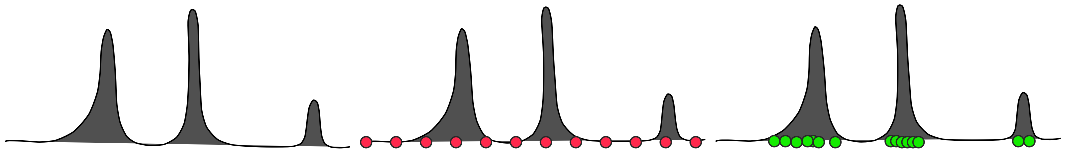


Figure 20: (*Left*) A function $f(x)$ can has many *peaks*. There might not be a single importance sampling density $p(x)$ that can capture regions where the function $f(x)$ has large values. (*Middle*) If samples (in red) are chosen uniformly, the variance will be high, because we oversample regions where the function is low (dark regions) and undersamples regions where the function is high (bright regions). (*Right*) If appropriate sampling density is used, we take many more samples (in green) in regions where the function has high values and thus reduce the variance.

variance reduction. The examples in the figure demonstrate that inappropriate sampling density can increase variance, which can even become infinite.

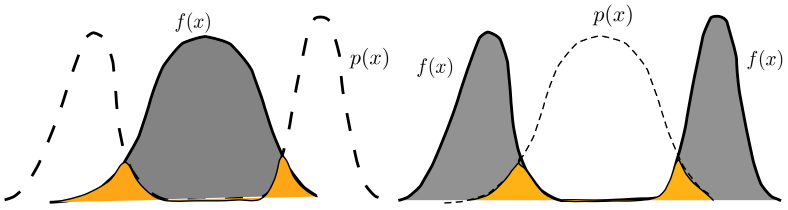


Figure 21: Bad choice of importance sampling density. The sampling density does not match the shape of the function $f(x)$ we want to evaluate. Only a small portion of the regions in the density function $p(x)$ overlap (in orange) the non-zero parts of the function. A bad choice of importance sampling density will **increase** variance and not reduce it.

## 7.2 Filtered Importance Sampling For Area Lights

Previous sections have described in great detail how to apply filtered importance sampling for infinite (hemispherical) lights. A small extension could be used for filtered importance sampling for textured area lights. This is an alternative approximation to what was described Section 5.3.

Recall that each sampled ray has a solid angle:

$$\Omega_s = \frac{1}{Np(\theta,\phi)}.$$

When the intersection with the sampled ray is found, we can approximate the area on the surface of the hit object by looking at the distance to the hit object and the solid angle of the ray:

$$A(x_i) \approx \frac{\|x_i - x\|^2 \cdot \Omega_s}{\cos\theta_i}. \tag{72}$$

For the above to hold true, we assume that the cross-sectional area is locally flat (Figure 22). Now that we have the estimated cross-section of the intersection $A(x_i)$, we can estimate the mipmap level $l$ based on this area:

$$l = \frac{1}{2}\log_2\left(\frac{A(x_i)}{A_{\text{pixel}}}\right) = \frac{1}{2}(\log_2 A(x_1) - \log_2 A_{\text{pixel}}) \tag{73}$$

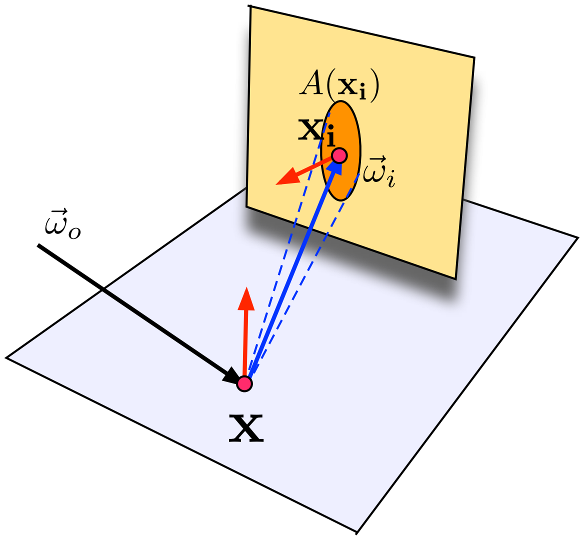


Figure 22: Cross-sectional footprint of ray intersection.

where $A_{\text{pixel}}$ is the object space area covered by one pixel. It is used to convert areas from object space to texel space. We can use this formula to filter the texture on area lights when using multiple importance sampling. It is important to recognize that this is a crude approximation. When part of the ray footprint is outside the textured area, the light contribution will be underestimated and wrong. Still, this approximation gives plausible results.

## 7.3 What About Visibility?

We have seen in Section 2.3.5 that the idealMonte Carlo estimator should be

$$\hat{L}_o = \frac{1}{N}\sum_{i=1}^{N}\frac{L_i B V}{p(\omega)}.$$

So far, we have been focusing on sampling from either the lighting, BRDF or a combination of sampling strategies using MIS. However, several recent sampling algorithms explicitly compute and sample an approximation of the product between the lighting and BRDF. Two-stage importance sampling [CETC06] and quadtree-based product sampling [CJAMJ05, CAM08b] hierarchically approximate a BRDF at a given point on the surface and combine it with the incoming light $L_i$. Some of these methods can be fairly costly or may require precomputed data structures for BRDFs. If BRDFs are spatially varying, some of these methods may not be practical since they would require too much storage for all the BRDFs in the scene.

We can take this further by adding visibility into the mix to lower the variance. Sampling from a triple product of lighting, BRDF and visibility, $LBV$, is difficult at best. Exact visibility would take a long time to pre-compute and would be expensive to store. We can use an approximate visibility $\bar{V}$, thus making the estimator $LB\bar{V}$. The problem with this approach is that approximate visibility $\bar{V}$ would have to be nonzero everywhere true visibility $V$ is nonzero. This brings us back to the initial problem, because in order to guarantee this condition is satisfied we would have to compute visibility in all directions.

On the other hand, we can use the estimator $LB\bar{V}$ as a control variate:

$$\hat{L}_o = \frac{1}{N}\sum_{i=1}^{N}\frac{L_iB\bar{V} - \alpha L_iB\bar{V}}{p(\omega)} + \alpha\int L_iB\bar{V}d\omega_i. \tag{74}$$

Remember that a control variate requires the function $f - g$ to be approximately constant. Even if visibility is crudely approximated, this condition can be satisfied. Clarberg and Akenine Möller [CAM08a] analyze the variance for this case and describe an algorithm for using approximate visibility:

- Create a compressed *visibility cache* using a compact bitwise representation stored at a sparse set of points in screen space.
- Compute a rough estimate of $L_iB\bar{V}$ using the approximations.
- Evaluate the difference between this approximation and the correct solution using Monte Carlo integration

The reader is directed to [CAM08a] for detailed discussion and implementation notes.

## 7.4 Resampled Importance Sampling

Efficiency of a Monte Carlo estimator depends on the expense of the sample evaluation versus the cost of drawing a sample from better densities. For example, if function is cheap to compute, but finding the importance density is computationally expensive, there is probably an advantage to using a simpler sampling density with more samples. In rendering, casting visibility rays can be expensive and oftentimes we still want to avoid tracing too many shadow rays.

Consider a situation where we have a glossy surface and we choose $N$ samples based on the BRDF density. We also know that the surface is very glossy and therefore the cone around reflected lobe is fairly narrow. We might be able to use less than $N$ visibility rays to approximate the light transport integral. We can do that by using *resampled importance sampling* [TCE05]. The idea is that from $N$ partial estimates (BRDF times lighting, $L_iB$) we only choose $M$ values for which we will compute the visibility.

More formally, resampled importance sampling is a generalization of importance sampling that permits unnormalized sampling densities or difficult to sample densities (in our case, visibility) denoted as $q$. In rendering, the best density $q$ would be $q \propto L_iBV$, but we can realistically at best only sample from another density $p$ that is proportional to $L_iB$. Instead of sampling from $q$, we generate a set of samples from a source distribution $p$ and weight these samples appropriately. Then, we *resample* these samples by drawing a single sample from them with probability proportional to its weight.

The basic algorithm proceeds as follows:

- Choose a set of sample $X_i$ from a known distribution $p$
- Associate a weight $w_i = \frac{q(X_i)}{p(X_i)}$ with each $X_i$, where $q$ is the desired (possibly unknown distribution)
- Generate the final samples $Y_i$ by sampling $X_i$ with a distribution proportional to $w_i$

If the weights $w_i$ are chosen to be $w_j = \frac{q(X_i)}{p(X_i)}$ then the resulting samples $Y_i$ will be approximately distributed to $q$. The processes of resampling is equivalent to filtering. In rendering applications, we use importance resampling as follows:

- Generate $N$ samples from some proposal distribution $p(x)$. This is as before, where we created samples from either lighting density, BRDF density or combined MIS density.

- Compute the weights (*e.g.*, luminance) of partial contributions ($L_i B$, but no visibility yet).
- Compute the discrete distribution from these weights.
- Chose $M$ samples from the above $N$ samples. These samples are chosen based on the importance density that we computed in previous step.
- Shoot shadow rays for these $M$ samples, add computed visibility to each sample and apply proper weighting (based on the probability with which each sample was chosen).

Note that although the desired target density $q(x)$ is unknown *a priori* because of the visibility, we never sample from it. We only need to be able to evaluate it and that is straightforward as long as as we can evaluate visibility $V$.

Resamples Importance Sampling (RIS) is better than importance sampling when:

- $q$ is a better importance sampling density than $p$.
- Computing proposals is much cheaper than computing actual samples.

RIS takes advantage of differences in the variance computation expense. More details and examples can be found in [TCE05].

## 8 Importance Sampling Framework at MPC

The Moving Picture Company recently developed a complete Image-Based Lighting solution in order to tackle the challenges of movies such as *Clash of the Titans*, *Harry Potter and The Deathly Hallows* and *The Chronicles of Narnia: The voyage of the Dawn Treader*. This project required us to render photorealistic creatures with highly reflective or wet surfaces, where fast and accurate specular and glossy reflections would be key to achieving the look. Alongside the creatures, we rendered a complete city environment, for which we would rely heavily on IBL for diffuse lighting.

Rather than augmenting traditional lighting techniques, our IBL tools became the standard solution and replaced classical lighting techniques in the majority of shots. This required us to tackle many problems, such as ensuring energy conservation and making ray-tracing of incredibly heavy scenes efficient by maximizing the bang for the buck of each ray traced. This section will cover some of these techniques and provide practical use cases of importance sampling and filtered importance sampling at MPC.

### 8.1 Image Based Lighting Constraints

Image Based Lighting allows the user to easily represent lighting coming from an almost infinite number of directional light sources. Different techniques can be used to compute the diffuse and specular contributions. Each of these techniques have different advantages in term of accuracy of the lighting and occlusion estimations as well as advantages in terms of computational cost. Amongst these techniques, the MPC pipeline uses a classic Dome Light, Environment Map Blurring, Importance Sampling and Filtered Importance Sampling. This allows the artist to choose a method based on its accuracy in the lighting, occlusion estimation, speed or memory consumption.

The use of Image Based Lighting raises different practical problems. This section describe some of the findings we made when using importance sampling for production purposes. We found that one important aspect raised by the introduction of this lighting technique in the MPC rendering pipeline was the difficulty to accurately estimate the incident lighting and at the same time ensure the quality of the shadows. Section 8.1.3 describes the ray-tracing solutions we developed at MPC to account for the variety of cases that our lighters encountered on the project Clash of the Titans.

Figure 23: Rendering of the Kraken character from Clash of The Titans. ©Warner Bros

Another aspect of IBL and Important Sampling is their ability to faithfully reproduce reflections. However, this functionality strongly depends on the choice of the BRDF. Section 4.1 presents how to sample the BRDF. In the next subsection, we describe one key aspect to account for in the choice of a BRDF when rendering feature films images.

### 8.1.1 Energy Conserving BRDFS and Albedo Pump-up

One important requirement of the Image Based Lighting at MPC was to ensure energy conservation in order to avoid or reduce "light leakage." Here, light leakage represents a loss of energy at near-grazing viewing directions which causes surfaces to appear darker near silhouette edges. When directly using energy-conserving BRDFs to compute reflectance, artists find a light-leaking or mirror reflection behavior around the grazing angles that is not intuitively expected. In the following, we compare the responses for three different BRDFs and their integration with Importance Sampling and Filtered Importance Sampling for computing the specular contribution of semi glossy surfaces.

The Ward BRDF [War92] has the advantage that its PDF is proportional to the BRDF and can be calculated at little extra cost. FIS using the Ward BRDF can give clean results even for a low number of samples. However, a direct implementation suffers from severe edge darkening, as visible in Figure 24.

The Ashikhmin-Shirley BRDF [AS00], a Phong-like anisotropic BRDF, also has an easily derivable PDF. However, as shown Figure 24, this BRDF also suffers from an edge darkening even though much less than the Ward BRDF.

Edward et al. [EBJ+06] presented a mathematical framework for enforcing energy conservation in a BRDF by specifying halfway vector distributions in simple two-dimensional domains. The Halfway-Vector Disk BRDF gives the best results in terms of energy conservation and faithfully reproduces the sharpening of reflections visible at grazing angles in real-world surfaces (Figure 24). However, this accuracy made the BRDF unsuitable for our needs. In production scenes, where traditional delta light sources are still used, the tendency of the reflection to be perfectly specular around grazing angles produced unattractive highlights.

In the end, we chose to use the Ashikhmin-Shirley BRDF, modified with an in-

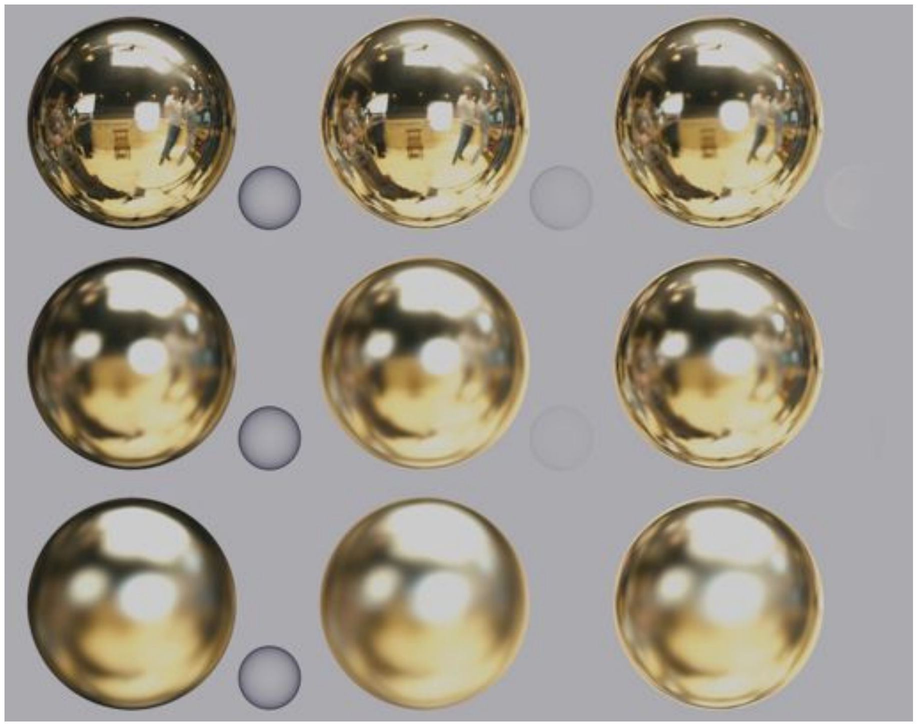


Figure 24: Comparison of the filtered importance sampling for the BRDF of Ward, Ashikhmin-Shirley BRDF and The Halfway Vector Disk BRDF. The grey spheres represent the response to a uniform grey environment. A sphere using an ideal energy conserving BRDF would not be visible; it would reflect all the light from the environment at each point of the sphere, so every point on the sphere would be the same color as the environment.

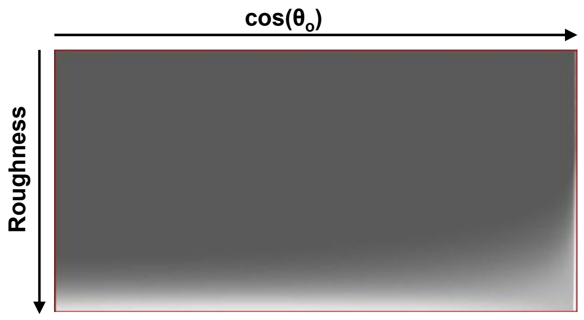


Figure 25: The inverse of the measured total reflectance is used to normalize our Ashikhmin-Shirley BRDF -here lowered in exposure by 2 for display purposes-.

house albedo pump-up based on the findings of Neumann et al [NNSK99]. While not completely physically accurate, this reduced the edge darkening to the point of being visually imperceptible. The pump-up also let us keep the blurry reflections at grazing angles desired by the artistic direction.

One simple approach to study and correct BRDF light leakage is to render a mirror ball within a constant white environment as presented in Figure 24. The albedo can be normalized by multiplying the BRDF by a normalization factor. This factor can be precomputed and stored in a texture, as the one in Figure 25. Here, the texture represents the inverse of total reflectance for every viewing angle and roughness value. The roughness is mapped from 0 to 1 to make the BRDFs interchangeable and the albedo pump-up is computed by

$$factor(\omega_o, roughness) = \frac{1}{\int_\Omega f(\omega_o, \omega_i, roughness) \cos\theta_i d\omega_i}. \tag{75}$$

Unfortunately, the addition of this arbitrary pump-up creates filtering artifacts when using FIS. Appearing as a ghosting effect, these artifacts are caused by filtering with an unmodified PDF. Experimental tests show that these are created by an overestimation of the filtering kernel around directions with a low PDF.

At first, we attempted to manually map filtering values that we compared visually to the correct result. While this gave acceptable results, it required long and fastidious experiments that had to be repeated for each BRDF.

We therefore modified Equation (63), replacing the PDF by $pow(pdf, \alpha)$. By ensuring that $\alpha < 1$, we successfully reduced the filtering for samples with a low PDF while preserving the correct filtering for samples with a high probability.

### 8.1.2 Dealing with Number of Samples

The quality of an image rendered with important sampling highly depends on the number of samples used to estimate the illumination integral (Equation (29)). A large number of samples usually provides accurate and noise free results, but it quickly becomes computationally prohibitive in complex scenes required for production. Here, the number of samples represents the amount texture lookups. In the case of ray-traced occlusion and inter-reflections, the sample count also represents the number of traced rays. While Filtered Importance Sampling allows us to use fewer samples than classical important sampling, it can still lead to prohibitive computations and it remains impor-

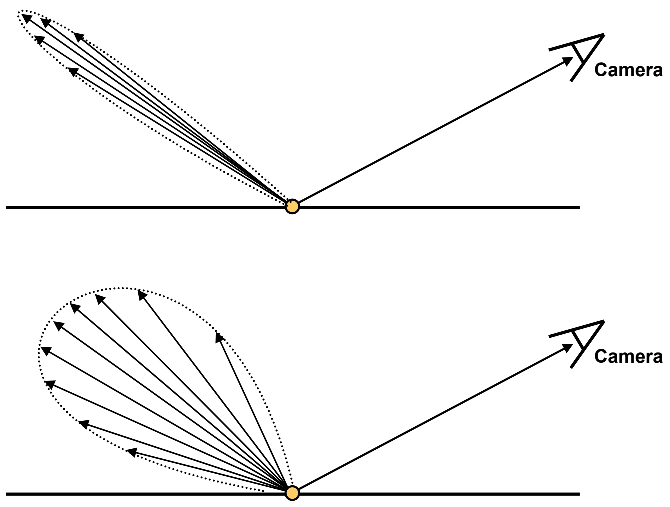


Figure 26: The number of samples needed to accurately estimate the incident lighting depends on the broadness of the specular lobe.

tant to try to use an optimal numbers of samples, especially in the case of ray-traced occlusion or inter-reflections.

**Adjusting the Number of Samples.** Depending on the shape of the BRDF and on the environment lighting, the number of samples should be adjusted. When considering no occlusion and lighting with a low frequency environment map, a small number of samples can efficiently estimate the incident lighting. On the contrary, using a low number of samples when lighting an object with high frequency incident lighting results in noisy images.

While the frequency of the environment lighting usually remains unknown, the number of samples needed to accurately estimate Equation (29) can also be related to the broadness of the BRDF lobe, as shown in Figure 26. Rendering mirror like reflections only require a few samples while rough surface reflections require a large number of samples. We mapped the number of samples to the shininess parameter of our BRDF to provide the artist with a simple but tweakable solution.

For a given roughness value, we rendered a reflective sphere with a different number of samples -ranging from 1 to 256-, as well as a reference sphere rendered with a large number of samples. The next step was to choose the sphere with the lowest number of samples that would effectively match the reference. We repeated this process for various values of roughness ranging from 0 to 1. This allowed us to plot the minimum number of samples required for each roughness value. We then fit a spline to the data to smoothen the results. The spline is stored in a look up table to minimize the computation for each pixel.

The roughness values of a character are often described by a texture map to allow rich specular variations over the model, such as wet/dry or more or less oily areas etc.. The interesting aspect of this mapping is that the sampling is automatically adapted to these roughness values. It significantly optimized the number of samples per pixel, especially if there are regions that are mirror like.

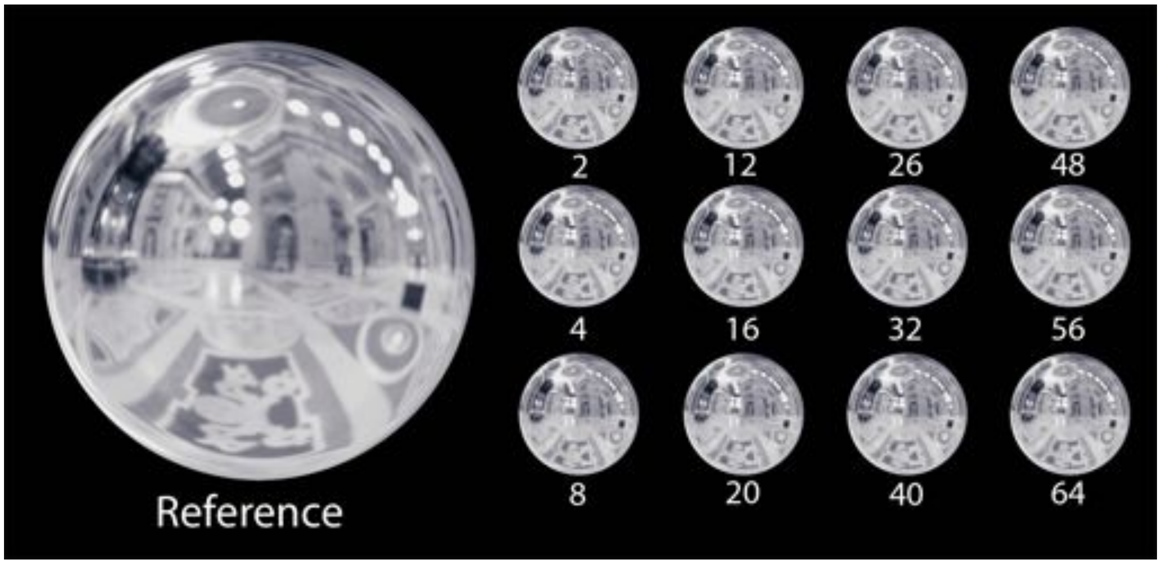


Figure 27: A given set of images is rendered with different sample counts and compared with a reference image to visually determine the necessary number of samples.

Figure 28 schematizes the increase of the number of samples with the increase of the material roughness. In most cases, when considering semi-glossy reflections without occlusion or inter-reflections, we found that values from 16 to 64 samples provided us with acceptable results in term of image quality and render time (red curve in Figure 28). For materials with strong roughnesses, which required high number of samples, a per material/scene tweaking was performed in order to optimize the render times.

**Adjusting the Number of Ray-traced Rays.** The computation of occlusion and inter-reflections for each sampled direction may be needed to accurately estimate the illumination integral. However, we found that computing the occlusion for the mapped number of samples was often too expensive and in many cases unnecessary. Depending on the scene geometry and its organization, we wanted to be able to adjust the amount of ray-tracing done (green curve in Figure 28). We introduced a percentage value that allowed us to only trace a certain number of sample directions while using interpolated occlusion or inter-reflection values for non ray-traced sampled directions.

There are multiple ways to interpolate the occlusion samples. A brute force technique is to directly interpolate the occlusion value for a given direction using the closest ray-traced directions. However finding these rays can be computationally expensive and can overcome the advantage of tracing fewer rays. Moreover, sorting the precomputed rays to accelerate the search can be time consuming and memory prohibitive and needs to be repeated for each shading point.

A more efficient method consists of projecting the occlusion function -estimated using the ray-traced sampled directions- into a hemispherical basis such as hemispherical harmonics [KG05] or hemispherical wavelets. An estimated occlusion value for the non-ray-traced directions is then obtained by evaluating the projected function in a new direction. Such methods give good interpolations when using enough coefficients which can, as a result, be time and memory consuming.

As our main requirement was to save memory and ensure speed, we simplify the problem and choose to sort the rays by quadrants and average the occlusion values

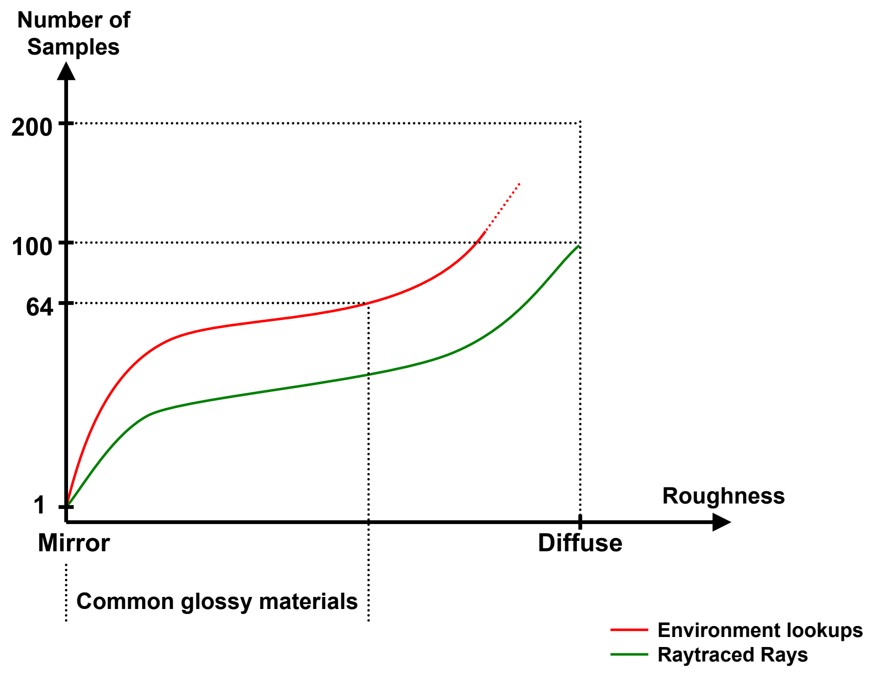


Figure 28: Schematic view of the mapping of the number of samples used at MPC.

per quadrant. The new directions will inherit the occlusion value of their respective quadrant. The quadrants are chosen around the direction of perfect reflection $\omega_r$ to ensure a good distribution of rays in all the quadrants (Figure 29).

This can be seen as a simplistic basis written:

$$\begin{aligned} B_1(\omega_i) &= \tfrac{(\alpha-1)(\beta-1)}{4} \\ B_2(\omega_i) &= \tfrac{(\alpha+1)(\beta-1)}{-4} \\ B_3(\omega_i) &= \tfrac{(\alpha+1)(\beta+1)}{4} \\ B_4(\omega_i) &= \tfrac{(\alpha-1)(\beta+1)}{-4} \end{aligned}$$

with $\alpha = sign(\omega_i.T)$, $\beta = sign(\omega_i.B)$, and $B$ and $T$ being random vectors constructed such that $(B, T, \omega_r)$ is an orthonormal basis.

The occlusion value $Occ(\omega_i)$ of a given direction $\omega_i$ is then efficiently estimated as follow:

$$Occ(\omega_i) = B_1(\omega_i) * Occ_1 + B_2(\omega_i) * Occ_2 + B_3(\omega_i) * Occ_3 + B_4(\omega_i) * Occ_4. \qquad (76)$$

Here, $Occ_1$, $Occ_2$, $Occ_3$ and $Occ_4$ are the respective average occlusion of each quadrant.

Even though this is a coarse approximation of the correct ray-traced occlusion, it gave us convincing results with a very small memory foot print. Furthermore, the simplicity of the projection made it easy to implement and provided us with quick but acceptable estimates of the occlusion values. When using this method, small artifacts, such as light leaks, could be visible for broad specular lobes but did not significantly change the overall look of the images in the very complex scenes we had in Clash of the Titans.

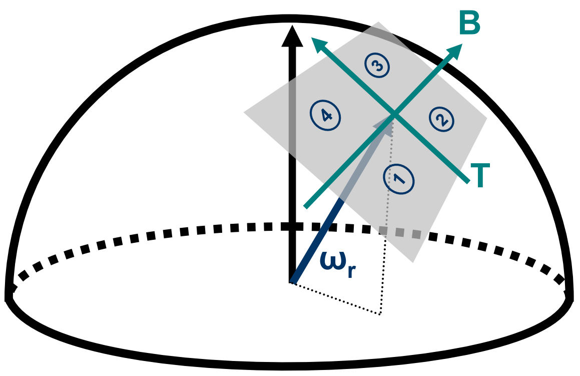


Figure 29: Each direction is located in a quadrant around the direction of the mirror reflection $\omega_r$.

Given an array of sample directions $\omega_i[]$, and the direction of the mirror reflection $\omega_r$, the following code snippet shows how to quickly estimate the visibility of the untraced rays.

```
vector randomVector = normalize(vector(random(),random(),random()));
vector T = randomVector^wr ;
vector B = wr^T;
color visibilityQuadrant[4] = {0,0,0,0};
int nSampleQuadrant[4] = {0,0,0,0};
int inQuadrant[4] = {0,0,0,0};

//Raytraced samples
for(int i=0; i<nRaytracedSamples; i+=1){
    alpha = sign(wi[i].T);
    beta = sign(wi[i].B);

    visibility = rayTraceVisibility(wi[i]);

    inQuadrant[0] =  1/4*(alpha-1)*(beta-1);
    inQuadrant[1] = -1/4*(alpha+1)*(beta-1);
    inQuadrant[2] =  1/4*(alpha+1)*(beta+1);
    inQuadrant[3] = -1/4*(alpha-1)*(beta+1);
    for(j=0; j<4; j+=1){
nSampleQuadrant[j] +=  inQuadrant[j];
visibilityQuadrant[j] +=  inQuadrant[j]*visibility;
    }
    //do the other computations
}

for(int j = 0; j<4; j+=1 ){
```

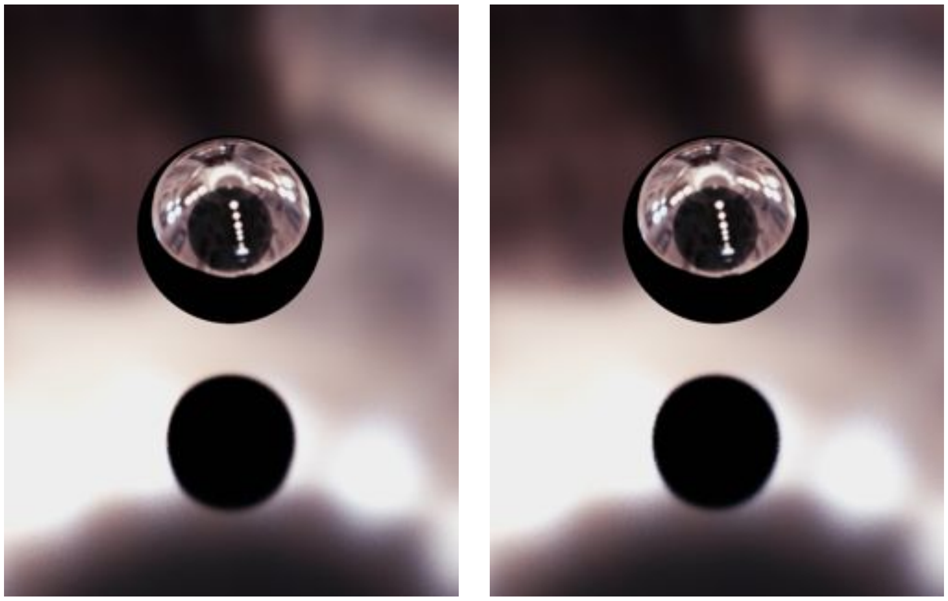

(a) (b)

Figure 30: Images rendered with 64 samples for the environment lookups with (a) 100% of traced rays for occlusion and (b) 60%.

```
        visibilityQuadrant[j] /= nSampleQuadrant[j];
    }

    //Non raytraced samples
    for(int i=nRaytracedSamples; i<nSamples; i+=1){
        alpha = sign(wi[i].T);
        beta = sign(wi[i].B);
        color visibility =  visibilityQuadrant[0]* 1/4*(alpha-1)*(beta-1) +
                            visibilityQuadrant[1]*-1/4*(alpha+1)*(beta-1) +
                            visibilityQuadrant[2]* 1/4*(alpha+1)*(beta+1) +
                            visibilityQuadrant[3]*-1/4*(alpha-1)*(beta+1);
        //do the other computations...
    }
```

In some cases, a quadrant may not contain any traced rays and would not store any estimation of the visibility. One solution is to trace the first ray in the quadrant and use its visibility value as an estimate of the whole quadrant's visibility. We found this solution expensive in practice as it adds a significant amount of conditionals and computations. Therefore, we chose a coarse approach and replaced the quadrant visibility by the averaged visibility value. Even though this is a very approximative solution, it showed acceptable behavior in practice.

Figure 30 shows that, in this test case, a reduction of the number of traced rays from 100% to 60% does not significantly increase the noise visible in the resulting image.

The next section presents other methods and tweaks we developed at MPC to efficiently use ray-tracing and Importance Sampling within the production constraints of Clash of the Titans.

### 8.1.3 Raytracing Strategies in Clash of the Titans

While FIS gives us good results and an acceptable render time when not computing the visibility term, these techniques became computationally and memory prohibitive when using ray-tracing to compute occlusion and inter-reflections. Even the interpolation technique in the previous section is insufficient at reducing the computational footprint. This section describes common methods which make possible the use of important sampling and ray-tracing within a production environment.

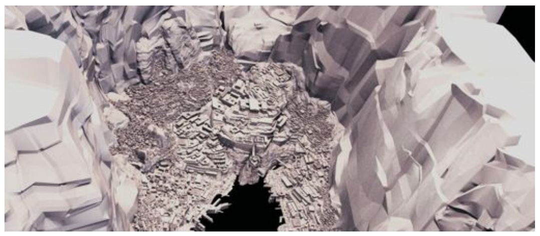
MPC has long favored Image-Based Lighting techniques over traditional point-source and convolved environment approaches for the superior complexity and richness of the resulting lighting. Clash of the Titans presented a unique challenge to MPCs lighting and shading team: a full 3D city-harbor environment being attacked by a 350feet-tall sea monster. The city environment consisted of thousands of individual buildings and props, totaling millions of polygons, while the Kraken itself was just as complex -dozens of subdivision surfaces represented by cages of over 7 million polygons-. All this had to be lit using Image Based Lighting and be reflected in the water surface of the bay, while the Kraken itself required complex glossy and specular self-reflections.

Despite significant performance improvements in recent versions, ray-tracing in Pixar's RenderMan is still an expensive operation. There are two ways to speed up the ray-tracing process: 1) shoot less rays, or 2) reduce the computation that results from each ray hit.

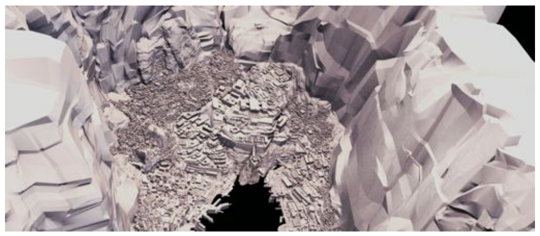
**Filtered Importance Sampling with Occlusion Caching and Interpolation.** As shown in the images below, Filtered Importance Sampling allowed us to dramatically reduce the number of rays required to sample the environment map effectively. Rather than use a single Image Based light source to illuminate the scenes, the initial image was split into different sectors to provide artists with full control over the lighting. In all, there were 6 individual IBL sources used to light the scenes. This required that we modify our shaders to cache the interpolated occlusion and importance sampling intermediaries for each sampled direction, such that the results could be re-used for each IBL source rather than tracing more rays. These results were also shared with reflection cards -simple textured polygonal objects hand-placed by lighters to add shot-specific reflections -which could be either occlusive or additive depending on the needs of the lighters-.

**Radiance/Irradiance Caching.** For several years, MPC has used radiance caching techniques similar to those described in [SRKG07] to avoid expensive shading computations for ray hits. Since we use RenderMan's point-based color bleeding solution [Chr09] on everything we render, we already have suitable irradiance pointclouds to be read for shading of reflections. Although these did not contain specular reflections, for the vast majority of cases the difference was not noticeable.

The use of Importance Sampling made ray-tracing for diffuse IBL of the city environment feasible, but it was still too slow to calculate on a per-frame basis. We could not bake the illumination into shared irradiance caches since the client required a unique lighting direction for every shot, and baking the entire city for every shot would be impractical. Thus, we wrote a tool to bake multiple key frames from the shots camera move, then stitch the resulting pointclouds together to create an irradiance pointcloud that covered the entire length of the shot. The shadows and specular reflections of the key light on the city were calculated in the beauty render (using FIS to limit the number of samples), so the environment lighting and color bleeding only needed to be rebaked relatively infrequently. Many shots were only baked once.



(a) infinite trace distance - 3 hours



(b) 60 meters - 32 minutes

Figure 31: The comparison of two renders with a different trace distance shows that reducing the trace distance does not significantly affect the look of the final image. ©Warner Bros.

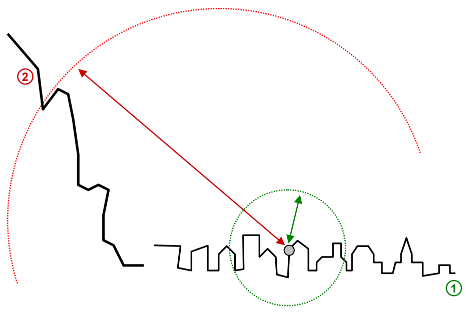


Figure 32: In Clash of the Titans, separate ray traces are performed with respective trace sets and trace distances.

**Trace Distance and Trace Sets.** A common technique for accelerating ray tracing is to limit the distance rays are traced for intersections before giving up and returning an unoccluded result. This is especially important for massive models, such as the city environment, since searching only a small part of the scene reduces that amount of cache thrashing. Here, the trashing is caused by reloading geometry that does not fit entirely in main memory.

For the particular case of lighting the city however, we required shadowing from the cliffs that surrounded the city in order to get the correct lighting result (Figure 31). The solution was to do two separate ray traces, one short distance against the high-resolution model for local inter-building occlusion, and one long distance against a very low-resolution proxy model of the cliffs for environmental shadowing (Figure 32). The importance sampled ray directions were shared between the two traces, and rays were not retraced for the cliffs if they were already occluded by the city.

**Ray-tracing Proxy Objects.** Even using when Filtered Importance Sampling to limit the number of samples, ray-tracing occlusion and specular reflections on a $\tilde{7}$ million polygon subdivision surface like the Kraken, presented in Figure 33, was still impractical. This was due to the computational cost and memory requirements. In order to make this feasible, we used a technique similar to that described in [TL04].

While subdivision surfaces were used for the beauty render, the traceable objects were in fact the original poly cages without any subdivision, thus giving much better performance both in memory usage and render time. In order to avoid intersections where the subdivision surface and the poly cage overlapped, we attached a second copy of the vertex positions to the subdivision surface, changing the way RenderMan interpolated the data such that the shaders would know where the polygonal cage was for each subdivision surface shading point, and could trace rays from this surface.

In instances where this still was not enough, we used the same technique but tracing against a low-poly representation of the Kraken ($\tilde{2}$ million polygons).

## 8.2 Quick Implementation of Iridescence and Color Shifts with Filtered Importance Sampling

The iridescence is an optical phenomenon commonly visible on multi-layered materials such as pearls, butterflies or snake skin. It consists in light wave interferences and diffraction on multiple thin films.
Different approaches can be used to either accurately estimate such phenomenon or to imitate it in a simpler way. The equations governing the iridescence phenomenon are complex and are not directly adaptable to the FIS theory because the importance sampling of such a BRDF is not straightforward. This section presents a simple and experimental approach to quickly imitate iridescent reflections when using Filtered Importance Sampling.

One key idea is to show the flexibility and simplicity of the FIS algorithm, even for reproducing complex phenomena such as iridescence (see Figure 34). The two important aspects of the iridescence phenomenon to take into account are the shifting of the BRDF lobes and the very remarkable change of color depending on the viewing angle and light incident angle.

**Shifting BRDF Lobe.** Iridescent materials commonly present off-specular reflections, such as retro reflection. The Lafortune BRDF [LFTG97] or the Halfway Vector Disk BRDF offer a good, controllable solution to this particular problem. However, note that comparable results can be obtained by shifting the BRDF lobes for other reflectance models, such as Ashikhmin-Shirley or Ward.

**Color Variation.** In order to provide the artist with a simple and editable iridescence effect, we chose to control the change of color using a user defined ramp of color such as the one presented in Figure 35. This color variation acts like a Fresnel term and multiply the environment contributions depending on the angle between the incident light direction and the normal, $\theta_i$. The interesting aspect of this method is its adaptability to the filtering scheme presented in Section 5. Note that in this case the distortion factor is equal to 1 when considering a ramp defined in $\theta_i$. However, the distortion needs to be taken into account when considering ramps defined in $cos(\theta_i)$.

One future extension of this method for imitating iridescence in FIS will be to take into account the anisotropic behavior of certain iridescent materials, such as snake scales, by considering color ramps defined in $\theta_i$ and $\phi_i$.

## 8.3 Sampling Strategies

One of the most commonly used algorithm for generating quasi-random sequences is the Hammersley algorithm (presented in Section 3), which is used in the FIS technique. One property of this technique is that it always generates the same sequence of quasi-random numbers for the same set of input parameters and provides an optimal deterministic

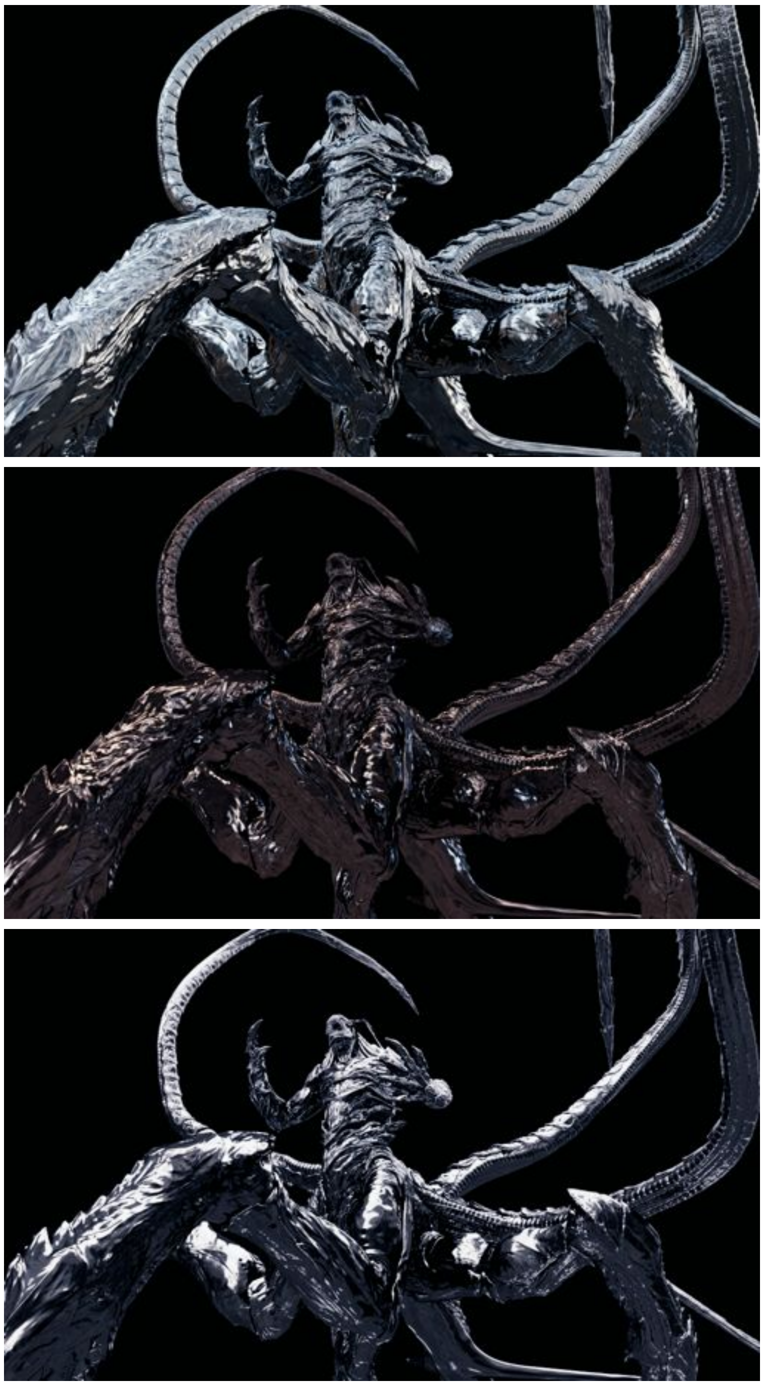


Figure 33: Renderings of the Kraken creature within different lighting environments. ©Warner Bros.

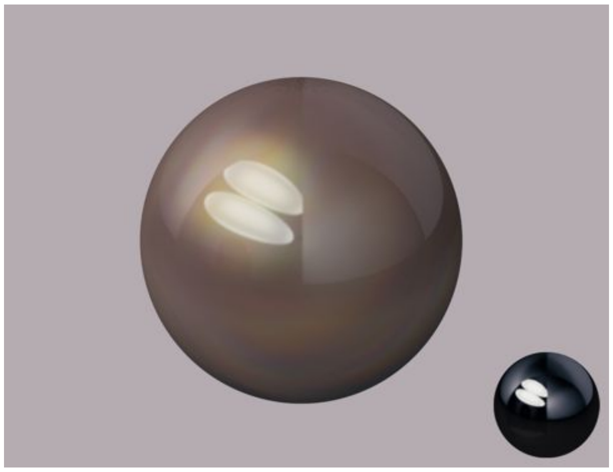


Figure 34: Iridescence and Filtered Importance Sampling. The colored reflection of the main sphere is a combination of mirror reflection and our iridescent reflection. The sphere in the lower right is a chrome reflection.

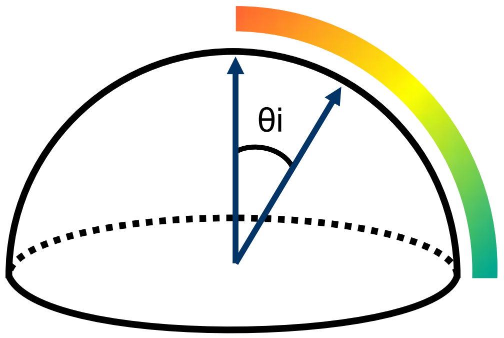


Figure 35: Example of ramp defining the color variations of the iridescence effect: the sphere in the lower right shows a chrome reflection, the colored reflection of the main sphere is entirely due to iridescence.

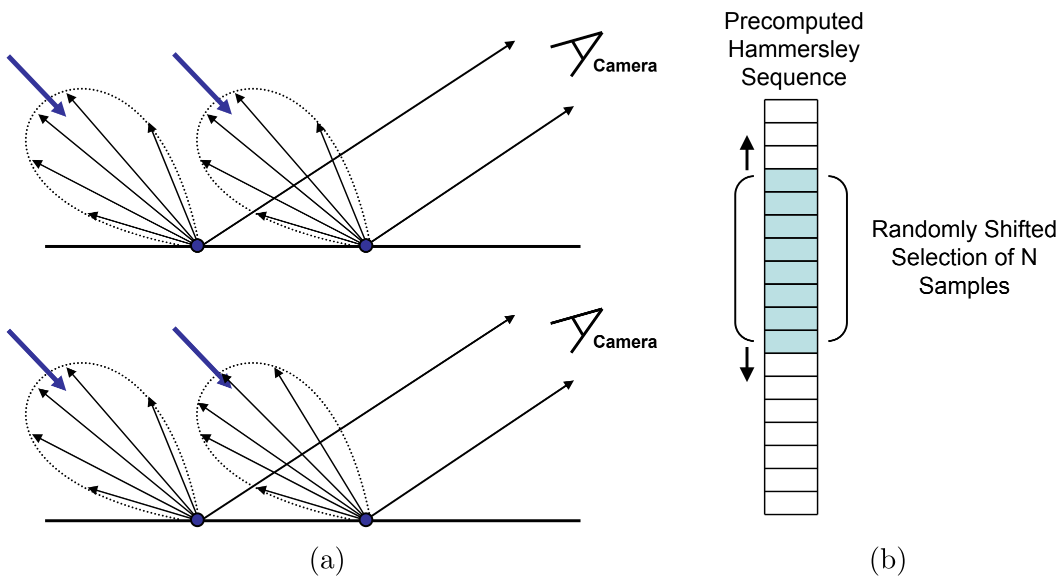

Camera
Camera
Precomputed Hammersley Sequence
Randomly Shifted Selection of N Samples
(a)
(b)

Figure 36: (a) Top: A quasi-random number sequence tends to generate spatial aliasing. Bottom: This can be avoided by using randomization algorithms. (b) The sequence selection is randomly shifted to avoid banding artifacts.

sequence. However, this deterministic behavior tends to produce spatial aliasing as explained by Figure 36. Reusing the same sequence of randomly distributed points to sample a function combined with a low number of samples, introduces a visual artifact known as banding or "chandelier effect" as seen in Figure 37(c). As explained by Colbert et al. [CPK06], in the case of Filtered Importance Sampling, this artifact is avoided as by blurring the environment map for each sample. However, the FIS technique was originally developed in the context of a GPU pipeline. Here, reflection occlusion and inter-reflection using ray-tracing are not commonly taken into account in the rendering equation (see Figure 37(a)). On the contrary, in production environments we need to capture these effects and the banding becomes a problem for which a solution is required.

One method to avoid these artifacts consists of using randomization methods. These methods, such as the Cranley-Patterson rotations or Random Digit Scrambling, "randomize" quasi-random sequences while keeping their low-discrepancy properties. A more complete description and comparison of these methods is presented in [KK02].

We use a simpler, fast to implement, technique at MPC: we store a large number of pre-computed quasi-random numbers using the Hammersley algorithm and shift the sequence used per shading point by offsetting the entry in the table by a random number. As visible in Figure 37(d), this method introduces some noise back in the frame. However, the results are considerably better than a purely random solution (Figure 37(b)) and avoid banding artifacts. Furthermore it is often better than other stochastic sampling strategies (nrooks,jittered, etc...) as it is still conserving the low discrepancy properties. An interesting aspect of our method is that one can balance between the banding and noise by controlling the amplitude of the random shifting. For a number of samples $N$, we found that using shifting amplitudes in between $N/2$ and $N$ gave us an acceptable balance. Note that the randomization method at MPC is a field that we will test and research more in the future.

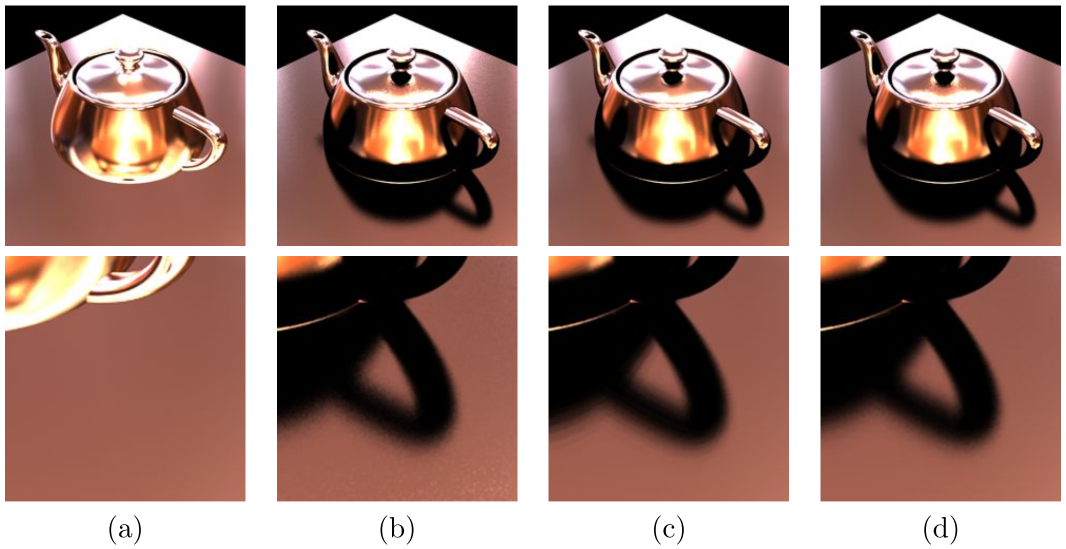


Figure 37: Comparison in between different sampling strategies. 16 samples are used to sample the BRDF. (a) FIS and Hammersley sampling with no occlusion, (b) IS and N-rook sampling with ray traced occlusion, (c) FIS and Hammersley sampling with ray traced occlusion, (d) FIS and Hammersley shifted sampling with ray traced occlusion.

## 9 Conclusion

Importance sampling is an effective framework for efficiently evaluating the illumination integral with only a limited amount of pre-computation. The approach provides a means to simulate light reflectance from a broad variety of materials with a relatively simple trade off between visual fidelity and the number of samples used in the computation.

The Filtered Importance Sampling (FIS) extension provides a practical solution for production rendering of both offline visual effects and real-time visualization. However, the approach does not work well for all situations, such as diffuse reflections. We have discussed the appearance of the FIS artifacts, which allows users to identify and diagnose problems associated with undersampling. For other cases, such as glossy reflection integration, FIS can provide smooth, deterministic results with a relatively small number of samples.

Multiple Importance Sampling (MIS) can further reduce the number of samples necessary for visually acceptable results since both the lighting and material functions guide the integration evaluation.

The importance sampling simulation algorithms can be implemented in most rendering systems as a single shader and thus easily integrate into most production pipelines. We have examined practical issues associated with various rendering systems and how they have been addressed in several production environments.

## 10 Acknowledgements

Thanks to ImageMovers Digital and Moving Picture Company for allowing the preparation and presentation of this course. A special thanks to Christophe Hery for many useful conversations and his code base for testing out various practical issues within RenderMan. Also, special thanks to Davy Wentworth and Doug Epps for their guidance and help in preparing the course.

Thanks to Anders Langlands, Jean Colas Prunier, Douglas McHale and all the Moving Picture Company look-development team for their help and suggestions which made the use of importance sampling techniques possible on the project *Clash of the Titans*.

## References

[ARBJ03] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. In *Proc. of ACM SIGGRAPH 2003*, pages 605–612. ACM, 2003.

[AS00] Michael Ashikhmin and Peter Shirley. An anisotropic phong BRDF model. *J. of Graph. Tools*, 5(2):25–32, 2000.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[BL08] Brent Burley and Dylan Lacewell. Ptex: Per-face texture mapping for production rendering. *Comp. Graph. Forum*, 27(2):1155–1164, 2008.

[BM58] George Edward Pelham Box and Mervin E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[CAM08a] Petrik Clarberg and Tomas Akenine-Möller. Exploiting Visibility Correlation in Direct Illumination. *Comp. Graph. Forum (Proc. of EGSR 2008)*, 27(4):1125–1136, 2008.

[CAM08b] Petrik Clarberg and Tomas Akenine-Möller. Practical Product Importance Sampling for Direct Illumination. *Comp. Graph. Forum (Proc. of Eurographics 2008)*, 27(2):681–690, 2008.

[CETC06] David Cline, Parris K. Egbert, Justin F. Talbot, and David L. Cardon. Two stage importance sampling for direct lighting. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 103–114, June 2006.

[Chr09] Per H Christensen. Point based approximate color bleeding. Technical Report #08-01, Pixar, 2009.

[CIE31] CIE. *Commission internationale de l'Eclairage proceedings.* Cambridge University Press, 1931.

[CJAMJ05] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM Trans. Graph.*, 24(3):1166–1175, 2005.

[CK07] Mark Colbert and Jaroslav Křivánek. *GPU-based Importance Sampling*, pages 459–475. GPU Gems 3. NVIDIA, 2007.

[CPK06] Mark Colbert, Sumanta Pattanaik, and Jaroslav Křivánek. BRDF-Shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE Comp. Graph. Appl.*, 26(1):30–36, 2006.

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982.

[Deb05] Paul Debevec. A median cut algorithm for light probe sampling. ACM SIGGRAPH 2005 Poster, 2005.

[EBJ$^+$06] Dave Edwards, Solomon Boulos, Jared Johnson, Peter Shirley, Michael Ashikhmin, Michael Stark, and Chris Wyman. The halfway vector disk for BRDF modeling. *ACM Trans. Graph.*, 25(1):1–18, 2006.

[Hes95] Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, May 1995.

[HH64] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* Wiley, New York, N.Y., 1964.

[Ige99] Homan Igehy. Tracing ray differentials. In *Proc. of SIGGRAPH 1999*, pages 179–186. ACM, 1999.

[KC08] Jaroslav Křivánek and Mark Colbert. Real-time shading with filtered importance sampling. *Comp. Graph. Forum (Proc. of EGSR 2008)*, 27(4):1147–1154, 2008.

[Kel01] Alexander Keller. Strictly deterministic sampling methods in computer graphics. Technical report, mental images, 2001.

[KG05] Jaroslav Křivánek and Pascal Gautron. Radiance caching for efficient global illumination computation. *IEEE Trans. Vis. Comp. Graph.*, 11(5):550–561, 2005.

[KK00] Thomas Kollig and Alexander Keller. *Monte Carlo and Quasi-Monte Carlo Methods*, chapter Efficient Bidirectional Path Tracing by Randomized Quasi-Monte Carlo Integration, pages 290–305. Springer-Verlag, 2000.

[KK02] Thomas Kollig and Alexander Keller. Efficient multidimensional sampling. *Comp. Graph. Forum*, 21(3):557–563, 2002.

[KW86] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods.* John Wiley and Sons, New York, N.Y., 1986.

[Lam60] Johann Heinrich Lambert. *Photometry, or, On the measure and gradations of light, colors, and shade.* The Illuminating Engineering Society of North America, 1760. Translated by David L. DiLaura in 2001.

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *Proc. of ACM SIGGRAPH 1997*, pages 117–126. ACM, 1997.

[Mit96] Don P. Mitchell. Consequences of stratified sampling in graphics. In *Proc. of ACM SIGGRAPH 1996*, pages 277–280. Addison-Wesley, 1996.

[NNSK99] Lszl Neumann, Attila Neumann, and Lszl Szirmay-Kalos. Reflectance models by pumping up the albedo function. In *Machine Graph. and Vision*, pages 3–18, 1999.

[ODJ04] Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph.*, 23:488–495, 2004.

[PH04a] Matt Pharr and Greg Humphreys. Infinite area light source with importance sampling. http://www.pbrt.org/plugins/infinitesample.pdf, 2004.

[PH04b] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation.* Morgan Kaufmann, 2004.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. In *Comm. of ACM*, pages 311–317. ACM, 1975.

[SG69] Jerome Spanier and Ely M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems.* Addison-Wesley, New York, N.Y., 1969.

[SRKG07] Apurva Shah, Justin Ritter, Chris King, and Stefan Gronsky. Fast, soft reflections using radiance caches. In *Proc. of ACM SIGGRAPH Sketches 2007*, page 52. ACM, 2007.

[TCE05] Justin Talbot, David Cline, and Parris Egbert. Importance resampling for global illumination. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, pages 139–146, June 2005.

[TL04] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. *ACM Trans. Graph.*, 23(3):469–476, 2004.

[VG95] Eric Veach and Leonidas Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proc. of ACM SIGGRAPH 1995*, pages 419–428. Addison-Wesley, 1995.

[Wal05] Bruce Walter. Notes on the Ward BRDF. Technical Report PCG-05-06, Cornell University, 2005.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proc. of ACM SIGGRAPH 1992*, pages 265–272. ACM, 1992.

[Wil83] Lance Williams. Pyramidal parametrics. *Comp. Graph.*, 17:1–11, 1983.