Importance Driven Quasi-Random Walk Solution of the Rendering Equation

László Szirmay-Kalos, Balázs Csébfalvi

Department of Control Engineering and Information Technology, Technical University of Budapest

Werner Purgathofer

Institute of Computer Graphics, Vienna University of Technology

Abstract

This paper presents a new method that combines quasi-Monte Carlo quadrature with importance sampling and Russian roulette to solve the general rendering equation efficiently. Since classical importance sampling and Russian roulette have been proposed for Monte-Carlo integration, first an appropriate formulation is elaborated for deterministic sample sets used in quasi-Monte Carlo methods. This formulation is based on integration by variable transformation. It is also shown that instead of multi-dimensional inversion methods, the variable transformation can be executed iteratively where each step works only with 2-dimensional mappings. Since the integrands of the Neumann expansion of the rendering equation is not available explicitly, some approximations are used, that are based on a particle-shooting step. Although the complete method works for the original geometry, in order to store the results of the initial particle-shooting, surfaces are decomposed into patches.

1 Introduction

The main goal of computer graphics is to calculate the image that could be seen by a camera in a virtual world. This requires the calculation of the power reaching the camera through a given pixel from the radiance of the surfaces. The radiance of an arbitrary point is also a function of other point radiances, since these other points may illuminate the given point which may reflect a given portion of the illumination. This type of coupling is expressed by the rendering equation:

$$L(\vec{x}, \omega) = L^e(\vec{x}, \omega) + \int\limits_{\Omega} L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\omega', \vec{x}, \omega) \; d\omega',$$

Preprint submitted to Elsevier Preprint

where Ω is the hemisphere above point $\vec{x}$, $L(\vec{x}, \omega)$ is the radiance of the surface in point $\vec{x}$ at direction ω, $h(\vec{x}, -\omega')$ is the visibility function defining the point that is visible from point $\vec{x}$ at direction $-\omega'$, θ' is the angle between the surface normal and direction $-\omega'$, and $f_r(\omega', \vec{x}, \omega)$ is the bi-directional reflection/refraction function (figure 1).

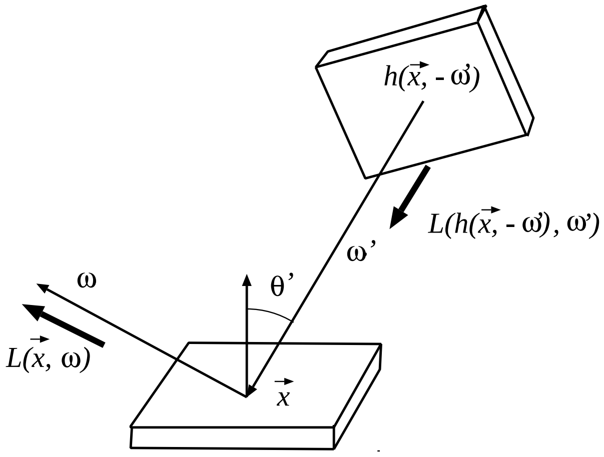

Fig. 1. Geometry of the rendering equation

To simplify the notations, the integral operator of the rendering equation is denoted by $\mathcal{T}$:

$$\int_{\Omega} L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\omega', \vec{x}, \omega)\ d\omega' = \mathcal{T}L(\vec{x}, \omega).$$

Thus the short form of the rendering equation is:

$$L = L^e + \mathcal{T}L. \tag{1}$$

In equation (1) the unknown function L appears on both sides. Substituting the right side's L by $L^e + \mathcal{T}L$ and assuming that $\mathcal{T}$ is contractive, we get the following Neumann series:

$$L = L^e + \mathcal{T}L = L^e + \mathcal{T}(L^e + \mathcal{T}L) = \ldots = \sum_{i=0}^{\infty} \mathcal{T}^i L^e.$$

The contractive property of $\mathcal{T}$ comes from the fact that a reflection or refraction always decreases the energy. The terms of this infinite Neumann series have intuitive meaning as well: $\mathcal{T}^0 L^e = L^e$ comes from the emission, $\mathcal{T}^1 L^e$ comes from a single reflection, $\mathcal{T}^2 L^e$ from two reflections, etc.

In order to understand how this can be used to determine the power going through a single pixel, let us examine the structure of $\mathcal{T}^i L^e$ as a single multi-dimensional integral for the $i = 2$ case:

$$(\mathcal{T}^2 L^e)(\vec{x}_1, \omega) = \int_{\Omega_1}\int_{\Omega_2} w_1 \cdot w_2 \cdot L^e(\vec{x}_3, \omega'_2)\ d\omega'_2 d\omega'_1.$$

where

$$\begin{aligned} \vec{x}_2 &= h(\vec{x}_1, -\omega'_1), \\ \vec{x}_3 &= h(\vec{x}_2, -\omega'_2) = h(h(\vec{x}_1, -\omega'_1), -\omega'_2), \end{aligned} \tag{2}$$

and the weights are

$$\begin{aligned} w_1 &= f_r(\omega'_1, \vec{x}_1, \omega) \cdot \cos\theta'_1, \\ w_2 &= f_r(\omega'_2, \vec{x}_2, \omega'_1) \cdot \cos\theta'_2. \end{aligned} \tag{3}$$

Thus to evaluate the integrand at point (ω'_1, ω'_2), the following algorithm must be executed:

(1) Surface point $\vec{x}_2 = h(\vec{x}_1, -\omega'_1)$ — that is the point which is visible from $\vec{x}_1$ at direction $-\omega'_1$ — must be determined. This can be done by sending a ray from $\vec{x}_1$ into direction $-\omega'_1$ and identifying the surface that is first intersected.
(2) Surface point $\vec{x}_3 = h(h(\vec{x}_1, -\omega'_1), -\omega'_2)$ — that is the point visible from $\vec{x}_2$ at direction $-\omega'_2$ — is identified. This means the continuation of the ray tracing at direction $-\omega'_2$.
(3) The emission intensity of the surface at $\vec{x}_3$ in the direction of ω'_2 is obtained and is multiplied with the cosine terms and the BRDFs of the two reflections.

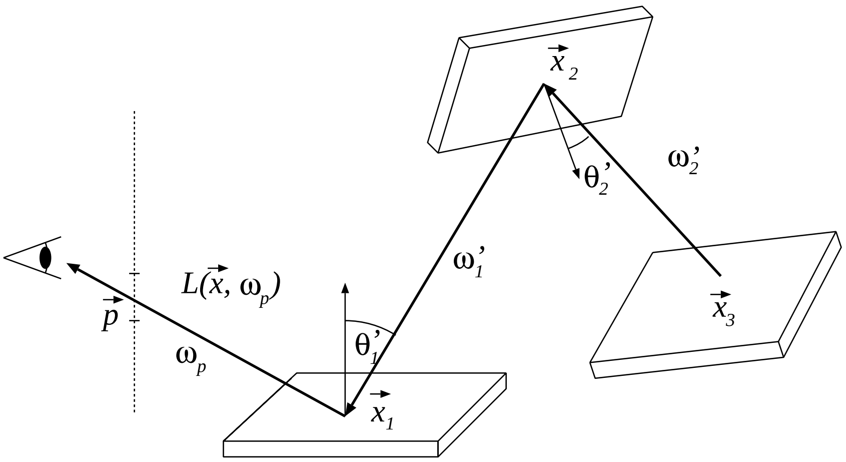

Fig. 2. The integrand of $\mathcal{T}^2 L^e$ is a two-step gathering walk

This algorithm can easily be generalized for arbitrary number of reflections. A ray is emanated recursively from the visible point at direction ω'_1 then from the found surface at ω_2', etc. until ω_n'. The emission intensity at the end of the walk is read and multiplied by the BRDFs and the cosine terms of the stages of the walk. This walk provides the value of the integrand at "point" $\omega_1', \omega_2', \ldots, \omega_n'$.

Note that a single walk of length n can be used to estimate the 1-bounce, 2-bounce, etc. n-bounce transfer simultaneously, if the emission is transferred not only from the last visited point but from all visited points.

The presented walking technique starts at the eye and *gathers* the illumination encountered during the walk. The gathered illumination is attenuated according to the cosine weighted BRDFs of the path.

1.1 Quasi-Monte Carlo Integration

For the sake of simplicity, assume that an s-dimensional function $f(\mathbf{z})$ needs to be integrated over the domain $[0,1]^s$. In order to guarantee that f is Riemann-integrable, f is assumed to be piece-wise continuous. For the normalized, s-dimensional integration domain $[0,1]^s$, the quasi-Monte Carlo approximation is:

$$\int\limits_{[0,1]^s} f(\mathbf{z})\; d\mathbf{z} \approx \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{z}_i). \tag{4}$$

Sample points $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M$ should be selected to minimize the error of the integral quadrature.

If the integrand f has finite variation in the sense of Hardy and Krause, then the error of the quasi-Monte Carlo quadrature can be bounded as stated by the Koksma-Hlawka inequality:

$$|\int\limits_{[0,1]^s} f(\mathbf{z})\; d\mathbf{z} - \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{z}_i)| \leq \mathcal{V}_{\mathrm{HK}} \cdot D^*(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M), \tag{5}$$

where $\mathcal{V}_{\mathrm{HK}}$ is the variation in the sense of Hardy and Krause and $D^*(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M)$ is the star-discrepancy of the sample points [10]. The star-discrepancy is defined by

$$D^*(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M) = \sup_A |\frac{m(A)}{M} - V(A)| \tag{6}$$

where A is an s-dimensional sub-cube parallel to the coordinate axes and originating at the center, $V(A)$ is its volume, and $m(A)$ is the number of sample points inside this sub-cube.

According to this inequality, there are two orthogonal strategies to improve the quadrature. Either the variation of the integrand or the discrepancy of the sample points should be reduced. The first approach corresponds to *importance sampling*. The second requires the careful selection of the sample points, that leads to *low-discrepancy sequences* [10]. The discrepancy of these sequences and consequently the error can be in the order of $O(\log^{s-1} M/M) = O(1/M^{(1-\epsilon)})$.

1.2 Importance Sampling

Importance sampling is a well-known method of Monte-Carlo integration to reduce variance. The basic idea is to use a non-uniform distribution to find sample points, which places more samples where the function is large. More specifically it means that the integral is estimated in the following form:

$$\int_V f(\mathbf{z})\, d\mathbf{z} = \int_V \frac{f(\mathbf{z})}{p(\mathbf{z})} \cdot p(\mathbf{z})\, d\mathbf{z} = E\left[\frac{f(\mathbf{z})}{p(\mathbf{z})}\right] \approx \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{f(\mathbf{z}_i)}{p(\mathbf{z}_i)} \pm \frac{\sigma}{\sqrt{M}},$$

where $p(\mathbf{z})$ is a probability density in V, the $\mathbf{z}_i$ points are selected according to this probability density, and the variance σ is defined by

$$\sigma^2 = E\left[\left(\frac{f(\mathbf{z})}{p(\mathbf{z})} - E\left[\frac{f(\mathbf{z})}{p(\mathbf{z})}\right]\right)^2\right] = \int_V \left(\frac{f(\mathbf{z})}{p(\mathbf{z})} - E\left[\frac{f(\mathbf{z})}{p(\mathbf{z})}\right]\right)^2 \cdot p(\mathbf{z})\, d\mathbf{z}.$$

The probability density $p(\mathbf{z})$ should be selected to minimize the variance. As can be shown easily, the variance can be minimized if $p(\mathbf{z})$ is proportional to the integrand $f(\mathbf{z})$. Thus in Monte-Carlo integration it is worth applying probability densities that are large where the integrand is large and small where the integrand is negligible.

2 Previous work

Monte-Carlo random walk methods generate samples of the light propagation from the lightsources to the eye. Since a photon may bounce on surfaces several times until it reaches the eye, the propagation consists of several surface to surface interactions, called steps. The series of these steps composes the walk.

Random-walks can be classified according to the direction they follow the light. A random-walk can be initiated from the lightsources and be traced in the direction of the light (shooting), or it can be started at the eye position and traced in the direction opposite to the light (gathering). Shooting type walks are also called *light walks*, while gathering type walks are called *eye walks*.

The problem of these methods is that the probability that a shooting path finds the eye is zero for a pin-hole camera or very small if a non-zero aperture camera model is used, while the probability that a gathering random path ends in a lightsource may be very little if the lightsources are small.

Thus a random walk must be combined with a deterministic step that forces the walk to go to the eye or to find a lightsource. *Light tracing* connects each bounce position to the eye deterministically [2]. *Bi-directional ray-tracing* methods start a shooting type and a gathering type walks and connect the bounce positions of the two walks. In a method proposed by Veach and Guibas [16] only the last positions of the two walks are connected. Lafortune [5], on the other hand, suggested that all positions visited during light-walks may be combined with the positions of eye-walks using a weighted sum.

In order to reduce the variance of the results, these methods incorporate some sort of importance sampling, which means that they use probability densities that places more samples where the integrand is large. However, the integrand of the rendering equation $L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\omega', \vec{x}, \omega)$ also depends on the unknown radiance function, thus this idea cannot be used directly. Instead, some approximation is required. The approximation applied in light-tracing and in bi-directional path tracing assumes that L is constant, thus the probability density of the new direction should be proportional to $\cos\theta' \cdot f_r(\omega', \vec{x}, \omega)$. If we select the proportionality ratio as 1, then the probability density $\Pr\{\omega'\} = \cos\theta' \cdot f_r(\omega', \vec{x}, \omega)$ will not be normalized, but integrates to a value less than 1. Since $\int_\Omega \cos\theta' \cdot f_r(\omega', \vec{x}, \omega)\, d\omega'$ also expresses the probability that no reflection occurs, this sub-critical probability density can be used as a termination criterion of the walk. Summarizing, this type of importance sampling uses only the BRDF and the cosine term to decide which the important directions are. Thus this becomes inefficient for scenes where the radiance distribution is not sufficiently uniform. This happens if the scene contains small and bright lightsources.

The other direction, called *direct lighting computation* [12], considers as important directions where the lightsources are visible. Thus their probability distributions are governed by the function $L^e(h(\vec{x}, -\omega'), \omega')$. Direct lighting methods can become inefficient when the BRDFs have high variation.

Realizing the complementary advantages of the two importance sampling techniques, Shirley et al. [11] used adaptive switching to select from the two alternatives, while Veach and Guibas [16] proposed a combined technique that generates the sampling distribution as a linear combination of several importance sampling distributions. One distribution could be responsible for emphasizing the radiance of lightsources, while the other could concentrate on the BRDF. However, the integrand of the rendering equation contains a product of the radiance and the BRDF, thus linear combination is still far from optimal.

Those algorithms that use the product of BRDFs and lighting for importance sampling are called *global importance sampling* methods. They are essentially two-phase procedures. In a preprocessing phase they build a data structure that guides the second phase to find important directions. These methods can be classified according to their incorporated data structure. Since the ray-space is 5-dimensional, it is straightforward to use a *5D adaptive tree* [6] that is similar to the well-known octree to store radiance information. Jensen, on the other hand, proposed the application of the *photon-map* as the basis of importance sampling [3]. In [14] the importance of the power exchange between two patches is represented by links connecting the respective patches.

Adaptive importance sampling methods neither require the non-uniform probability densities to be constructed in advance, nor simplify them taking only local properties into account, but converge to a desired probability density using the knowledge of previous samples. Three techniques are particularly important, which have also been used in rendering: *genetic algorithms* [7] the Metropolis sampling [9,17] and the VEGAS method [8,13].

Quasi-Monte Carlo techniques have also gained a lot of attention in rendering. These techniques have been first applied to solve the diffuse radiance equation by Keller [4], where the integrand was generally discontinuous and thus of infinite variation, which prohibited the theoretical justification of the superiority of quasi-Monte Carlo method over Monte-Carlo approaches [15]. However, the numerical evidence showed that quasi-Monte Carlo methods can slightly be better than Monte-Carlo techniques.

3 Motivation

This paper has multiple objectives. From a theoretical point of view it examines how low-discrepancy sequences can be used with methods developed for classical Monte-Carlo techniques, including importance sampling and Russian roulette.

Then, the paper aims at improving the previous techniques by proposing an algorithm that uses path tracing with fully exploiting importance sampling to solve the general rendering equation. The algorithm consists of two phases. In the preprocessing phase photon tracing (forward tracing) is executed to find an estimation of the radiance function. In order to store the estimated function, the surfaces are decomposed into patches and for each patch those directions are stored explicitly where photons have come from. In the second phase of the algorithm visibility ray-tracing walks (backward tracing) are executed for each pixel. The directions of this walk are generated by importance sampling based on the information gathered in the photon-shooting step.

During both the photon-tracing and the visibility ray-tracing steps, the sample points are generated from a deterministic low-discrepancy series rather than using a random distribution. Thus both steps use quasi-Monte Carlo integration instead of Monte-Carlo quadrature.

4 Importance sampling and Russian roulette with quasi-Monte Carlo methods

4.1 Application of importance sampling in quasi-Monte Carlo integration

When solving the rendering equation, usually directional integrals (or surface integrals in other formulation) should be evaluated. Thus to allow the application of low-discrepancy point sets, the integration domain should be transformed to the unit square. For example, when dealing with directions, we have to find a mapping $\omega = T(\mathbf{z})$ that projects the unit square to the surface of the sphere (or hemisphere) and use the following integration rule

$$\int_{\Omega} f(\omega)\, d\omega = \int_{[0,1]^2} f(T^{-1}(\mathbf{z})) \cdot \left| \frac{dT^{-1}(\mathbf{z})}{d\mathbf{z}} \right| \, d\mathbf{z}, \tag{7}$$

where

$$\left| \frac{dT^{-1}(\mathbf{z})}{d\mathbf{z}} \right| = \frac{1}{t(\mathbf{z})}$$

is the Jacobi determinant of the inverse mapping.

If the Jacobi determinant is large, then a small portion of the unit square is mapped onto a large region. Thus sample points that are uniformly distributed in the unit square will be quite rare in these regions. Alternatively, where the Jacobi determinant is small, the sample points will be dense. Considering this, the meaning of $t(T^{-1}(\mathbf{z}))$ is the *density* of sample points in the neighborhood of $\omega = T^{-1}(\mathbf{z})$. This has an illustrative meaning for the random case. If $\mathbf{z}$ is uniformly distributed random variable, then the probability density of $\omega = T(\mathbf{z})$ will be $t(\mathbf{z})$.

Mathematically, the solution of the rendering equation for a given point $(\vec{x}, \omega)$ requires the evaluation of the following multi-dimensional integral:

$$L(\vec{x}, \omega) = L^e + \mathcal{T} L^e + \mathcal{T}^2 L^e + \ldots =$$

$$L^e + \int \ldots \int \frac{w_1}{t_1} \cdot L^e(\mathbf{z}_1) + \frac{w_1}{t_1} \cdot \frac{w_2}{t_2} \cdot L^e(\mathbf{z}_1, \mathbf{z}_2) + \ldots d\mathbf{z}_1 d\mathbf{z}_2 \ldots \tag{8}$$

which can be estimated using formula (4) by evaluating the integrand in sample points and averaging the results.

Following the directions concluded from the Koksma-Hlawka inequality, the mappings should make the integrand flat — that is of low variation, or constant in the ideal case.

Looking at formula (8), this decision seems to be hard to make, since there are too many free parameters to control simultaneously. Fortunately, this solution can also be presented in the following recursive form:

$$L^e + \int \frac{w_1}{t_1} \cdot [L^e + \int \frac{w_2}{t_2} \cdot [L^e + \ldots] \ldots] \; d\mathbf{z}_1 d\mathbf{z}_2 \ldots \tag{9}$$

If we could ensure that each of the integrands of the form

$$\int \frac{w_i}{t_i} \cdot [L^e + \int \ldots] \; d\mathbf{z}_i$$

is constant (at least approximately), then the integrand of the single multi-dimensional integral will also be constant.

Thus this consideration leads us to the following quasi-Monte Carlo importance sampling method:

(1) A point from a $2M$-dimensional low-discrepancy set is generated.
(2) The tracing is started at the eye position and the surface visible in a pixel is found.
(3) At the found point $\vec{x}$, function $L \cdot \cos\theta' \cdot f_r$ is estimated and the estimation is inverted to determine T_1^{-1}. Using this inverse mapping the first two coordinates of the low-discrepancy set is transformed onto the directional hemisphere. These directions are used to continue the ray-tracing and the resulting colors are divided by the estimated value.
(4) In the subsequent recursive steps of ray-tracing, $L \cdot \cos\theta' \cdot f_r$ is estimated and inverted, then using the next higher dimensional coordinates of the low-discrepancy sequence, the new direction is determined.

Note that the importance should be proportional to a product of two functions: the estimated radiance L and the known, weighted BRDF which is $\cos\theta' \cdot f_r$.

In order to make this approach practically feasible, two problems need to be solved. We need effective methods to estimate radiance L for a given $\vec{x}$ and to invert the product of the radiance and the weighted BRDF.

4.2 Handling infinite-dimensional integrals using low-discrepancy sequences

Expansion, or walk methods require the evaluation of infinite-dimensional integrals. One way of attacking the problem is truncating the Neumann series, but this introduces some bias, which can be quite high if the scene is highly reflective.

Fortunately, there is another approach that solves the infinite-dimensional integration problem through randomization. In the context of Monte-Carlo integration, this approach is called the *Russian roulette* [1], but here a somewhat more general treatment is given that can also justify this approach for quasi-Monte Carlo quadrature.

The basic idea is very simple. The domain of integration is extended by additional variables on which a contribution indicator function is defined that will determine whether or not a higher order term is included in the quadrature (interestingly, in order to get rid of high-dimensional integrals, we increase the dimension of the integration). In order to compensate the missing terms in the integral quadrature, the really computed terms are multiplied by an appropriate factor. If the used contribution indicator is such that the domain where it is non-zero shrinks quickly, then the possibility of requiring very high dimensional integrals is rather low, which saves computation time. However, the integral quadrature will still be correct asymptotically.

After the transformation required by importance sampling, a term of the Neumann series has generally the following form

$$\int \ldots \int W(\mathbf{z}_1, \ldots, \mathbf{z}_n) \cdot L^e(\mathbf{z}_1, \ldots, \mathbf{z}_n) \; d\mathbf{z}_1 \ldots d\mathbf{z}_n, \tag{10}$$

where $W(\mathbf{z}_1, \ldots, \mathbf{z}_n) = w_1/t_1 \cdot \ldots \cdot w_n/t_n$ is the product of the weights and the densities of the importance sampling.

Let us extend this by a *contribution indicator function* $C(\mathbf{z}_1, r_1, \ldots, \mathbf{z}_n, r_n)$ that is 1 if a sample $\mathbf{z}_1, \ldots, \mathbf{z}_n$ should be taken into account in the integral quadrature and 0 if it should not. Usually this function is in separable form:

$$C(\mathbf{z}_1, r_1, \ldots, \mathbf{z}_n, r_n) = \prod_{i=1}^{n} c_i(\mathbf{z}_i, r_i).$$

Function $c_i(\mathbf{z}_i, r_i)$ can be defined by a new weight function $\xi(\mathbf{z}_i)$ in the following way:

$$c_i(\mathbf{z}_i, r_i) = \begin{cases} 1 & \text{if } \xi(\mathbf{z}_i) > r_i, \\ \\ 0 & \text{otherwise.} \end{cases}$$

Using this, the average "possibility" of continuing a walk at step i with value $\mathbf{z}_i$ is

$$\int_{r_i=0}^{1} c(\mathbf{z}_i, r_i)\ dr_i = \int_{r_i=0}^{\xi(\mathbf{z}_i)} dr_i = \xi(\mathbf{z}_i).$$

We can define the following function of variables $r_1, \ldots, r_n$:

$$I_n(r_1, \ldots, r_n) = \int \ldots \int C(\mathbf{z}_1, r_1, \ldots, \mathbf{z}_n, r_n) \cdot \tilde{W} \cdot \tilde{L}^e\ d\mathbf{z}_1 \ldots d\mathbf{z}_n, \quad (11)$$

where $\tilde{W}$ and $\tilde{L}^e$ are appropriate modifications of W and L^e, which can compensate the missing terms.

The integral of this function is

$$I_n^* = \int_{r_1=0}^{1} \ldots \int_{r_n=0}^{1} I_n(r_1, \ldots, r_n)\ dr_1 \ldots dr_n =$$

$$\int_{r_1=0}^{1} \ldots \int_{r_n=0}^{1} \int \ldots \int C(\mathbf{z}_1, r_1, \ldots, \mathbf{z}_n, r_n) \cdot \tilde{W} \cdot \tilde{L}^e\ d\mathbf{z}_1 \ldots d\mathbf{z}_n\ dr_1 \ldots dr_n =$$

$$\int \ldots \int \xi(\mathbf{z}_1) \cdot \ldots \cdot \xi(\mathbf{z}_n) \cdot \tilde{W} \cdot \tilde{L}^e\ d\mathbf{z}_1 \ldots d\mathbf{z}_n. \quad (12)$$

Obviously, I_n^* equals to the original integral of equation (10) if

$$\xi(\mathbf{z}_1) \cdot \ldots \cdot \xi(\mathbf{z}_n) \cdot \tilde{W} \cdot \tilde{L}^e = W \cdot L^e. \quad (13)$$

There are many possible selections of the contribution indicator and the $\tilde{W}$ and $\tilde{L}^e$ functions, that can satisfy this requirement, thus there are many different unbiased estimators.

A widely used selection is letting

$$\tilde{W} = 1, \quad \tilde{L}^e = L^e \ \text{ and } \ \xi_i(\mathbf{z}_i) = w_i/t_i.$$

which corresponds to continuing the walk after step i only if $w_i/t_i > r_i$. If importance sampling generates samples according to w_i, then w_i/t_i is constant and is equal to $\int w_i\ d\omega$, i.e. to the albedo of the found point. Thus this approach continues the walk with the "possibility" of the albedo. This can also be interpreted as using a mapping from the unit square of low-discrepancy samples to the domain of directions in such a way that certain low-discrepancy points do not correspond to valid directions but force the walk to be terminated.

4.3 Estimation of the radiance function

The unknown radiance function can be estimated using some information gained in an appropriate preprocessing step. This preprocessing step is basically a photon-tracing algorithm that shoots the energy of the lightsources into the scene and registers all surface-to-surface interactions. What we need is the determination from which directions a given patch can receive substantial light energy, no matter whether it comes from direct illumination or from multiple reflections.

For each patch, this information is stored in linked lists containing those directions from which photons have come into the given patch. Assume that the domain of directions ω' is decomposed into finite number of "pixels". The importance of these pixels is determined in the ray-tracing step, since it does not depend just on the incoming radiance but also on the viewing direction. Whenever a surface is hit, a so called *importance map* is generated by processing the linked list of the patch containing the intersection point, and different weights are assigned to the pixels. The resolution of the importance map depends on the number of the stored vectors and it can be set using a heuristic formula. In order to determine the pixel "intensities", the cosine weighted BRDF ($\cos\theta' \cdot f_r(\omega', \vec{x}, \omega)$) is evaluated for each stored direction ω'. Finally, the intensity of a pixel is the sum of the weights of the corresponding reflection directions.

4.4 Inversion of the importance

The inversion process should determine a mapping T that maps the importance map onto a unit square where the low-discrepancy points are located. This mapping has to map pixels of the importance map to areas that are proportional to the intensity of the pixel. We propose two different techniques for the implementation of this mapping. The first one is really two-dimensional, while the other handles the mapping approximately via a one-dimensional transformation.

4.4.1 *Mip-map based inversion*

This transformation method is based on a modified version of the classical idea of mip-mapping used to store textures. First, let us suppose that the importance map consists of just 4 pixels with values I_1, I_2, I_3 and I_4.

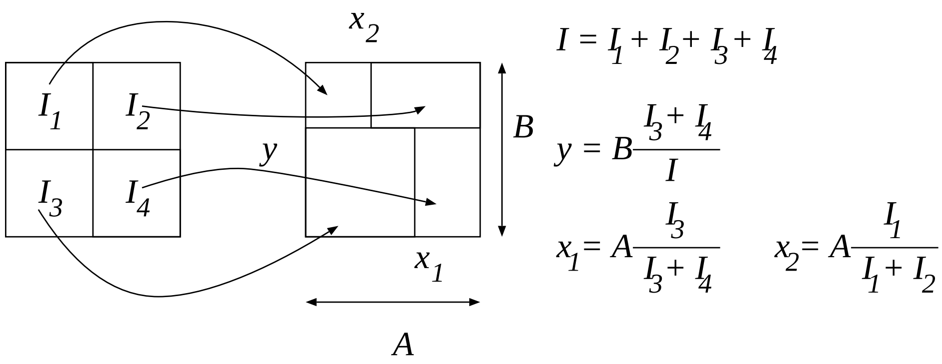

Fig. 3. Mip-map organization of the importance map

An appropriate transformation that decomposes a rectangle of size $A \times B$ to 4 subrectangles having area proportional to the four values is shown in figure 3. The inversion process — which generates a ϕ, θ point from a point in the rectangle — works as follows:

(1) A point is generated in the $A \times B$ rectangle.
(2) The $A \times B$ rectangle is decomposed into four subrectangles proportionally to the corresponding values in the importance map.
(3) The subrectangle that contains the given point is found.
(4) The point is projected from the $A \times B$ rectangle to the rectangle of size $[2\pi, \pi]$ by a linear transformation that maps the found subrectangle onto the corresponding pixel of the importance map.

The same idea can be used recursively if the importance map consists of $2^n \times 2^n$ pixels where $n > 1$. In the first step a mip-map scheme is built, which halves the resolution at each level. Having generated a point in the unit square and recursively applying the transformation of figure 3, the corresponding pixel is found in the importance map. The directional coordinates are then generated by a linear transformation which converts the found subrectangle to the pixel of the importance map.

4.4.2 *Linear inversion*

This inversion method takes advantage that the importance map is discretized, thus it can be considered as a one-dimensional, row-continuous array. The algorithm selects randomly or quasi-randomly a single pixel from this array proportionally to its value.

First a uniformly distributed value is generated either randomly or quasi-randomly in the range of $[0, I]$ where I is the sum of all values in the importance map. Then the importance map is scanned row by row and each importance value is added to a running variable. If this running variable is greater or equal to the generated random value, then the last visited pixel is selected. Here the direction is generated using uniform distribution on the pixel.

Note that this method is very similar to generating arbitrary random variables by the inversion of the cumulative distribution function.

5 The algorithm

The photon-shooting step shoots photons of the same power. Suppose that the power emitted by lightsource l is Φ_l, while the total emitted power is Φ_T. If the number of required photons is n, then we should shoot $n_l = n \cdot \Phi_l / \Phi_T$ number of photons from lightsource l. Having shot a photon, its walk is traced on different surfaces. The shooting and reflection directions are generated from an infinite dimensional Halton sequence. In order to calculate the reflection direction, the uniform distribution of a two-dimensional subspace of the Halton sequence is mapped onto the cosine weighted BRDF. Since the weighted BRDF is subcritical — that is, it integrates to a value less than 1 — the walk is terminated when the uniformly distributed point is transformed onto no real direction.

The following algorithm summarizes the photon-shooting step, where $H(i, j)$ stands for the pair of the next Halton points generated using the bases of the ith and the jth prime numbers respectively.

for each lightsource l **do**
    $n_l = n \cdot \Phi_l / \Phi_T$
    for $p = 1$ **to** n_l **do**
        Find starting point on lightsource l using $H(1, 2)$
        Find starting direction ω' using $H(3, 4)$ transformed
            by the directional distribution of the lightsource
        $i = 1$
        while not absorbed
            Shoot photon from **source** at direction ω'
            Find **destination** using the nearest intersection
            Add the direction of the **source**
                to the list of the **destination** patch
            Generate reflection direction ω using
                $H(3 + i \cdot 2, 4 + i \cdot 2)$ transformed by $\cos\theta' \cdot f_r(\omega', \vec{x}, \omega)$
            if not absorbed $\omega' = \omega$
            $i = i + 1$

endwhile
endfor
endfor

This phase is followed by the view-dependent ray-tracing step that uses direction vector lists to guide importance transformation. For each pixel, several rays are traced recursively. The point inside the pixel where the ray gets through the window is also determined from the transformation of a quasi-random distribution. The ray-tracing step is presented in the following:

for each pixel **do**
  $color = 0$
  Generate and sort Halton points
  for each pixel sample $i = 1$ **to** n_{sample}
    Find starting direction ω using $H(1, 2)$ transformed
      by the importance of location inside the pixel
    $color$ += RayTrace($\vec{eye}, \omega, i, 1, 1)/n_{\text{sample}}$
    endfor
  endfor
  Display pixel($color$)
endfor

The RayTrace subroutine is responsible for recursively following an eye-walk:

RayTrace $(\vec{eye}, \omega, i, level, atten)$
  Find the first intersection: $\vec{x}$, surface
  Generate Importance map $I[u][v]$
  $I = \sum_u \sum_v I[u][v]$
  if $(atten > minatten)$
    Generate reflection direction ω' and find importance
      $I[u^*][v^*]$ using $H(2 \cdot level + 1, 2 \cdot level + 2)$
    $r = \cos\theta'(u^*, v^*) \cdot f_r(\omega'(u^*, v^*), \vec{x}, \omega) \cdot I/I[u^*][v^*]$
    return $L^e(\vec{x}, \omega) + A_{\text{pixel}} \cdot r \cdot \text{RayTrace}(\vec{x}, \omega', level + 1, atten \cdot r)$
  else return $L^e(\vec{x}, \omega) + I \cdot \Phi_T/n$
end

The ω' reflection direction is taken from a uniform distribution on the selected pixel of the directional hemisphere. The density function of this conditional distribution is $1/A_{\text{pixel}}$, where the A_{pixel} is the area of the given pixel, thus the contribution of the reflected ray has to be multiplied by A_{pixel}. Note that the algorithm traces the rays until any further tracing would have no significant effect (the attenuation factor gets lower than a predefined minimum). When this happens rays are not followed any further, but the illumination is approximated from the importance map. Since the importance map is an approximation of the radiance, it can also be used to determine the radiance in earlier phases. This would result in an algorithm where the color of the ray would be generated as weighted sum of the color of the reflected ray and the sum of the importance map multiplied by the Φ_T/n energy of a photon. As the level of the recursion increases, the importance map could get higher weights.

6 Implementation

The proposed algorithm was tested with 100K photons and the images were generated taking 20 and 100 samples per pixel. In figure 4 the same scene is shown rendered with 20 samples per pixel using BRDF sampling and the proposed importance sampling respectively. Although the distribution of the variance is much uniform in the left image, the error is much lower in the right one, especially in those patches which have been hit by large number of photons in the first phase. For example, the variance on the boxes is rather small comparing to the left image. Figure 6 shows the images rendered taking 100 samples per pixel.

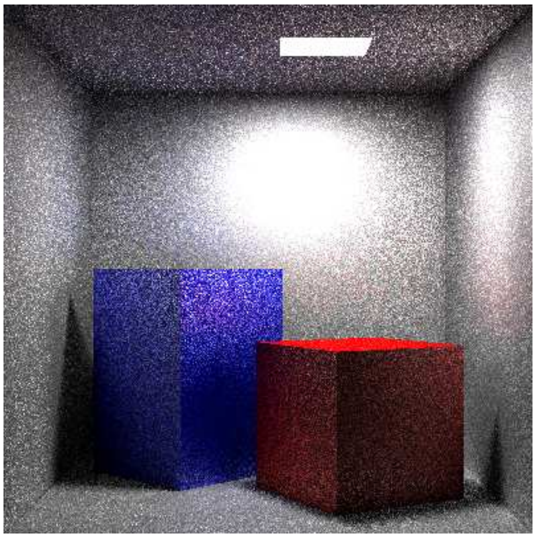

Fig. 4. BRDF based sampling with 20 samples per pixel

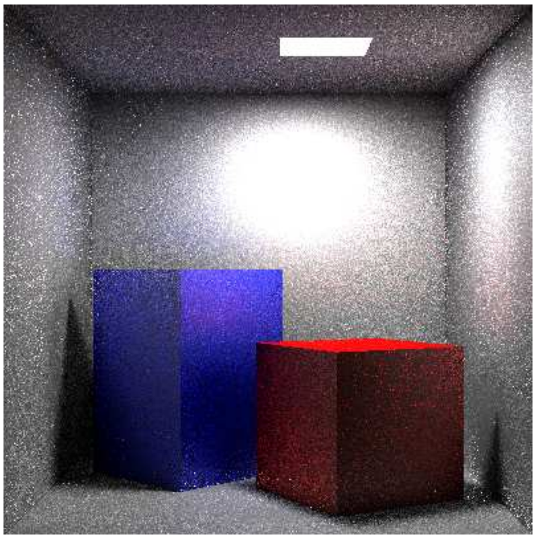

Fig. 5. Importance driven sampling with 20 samples per pixel

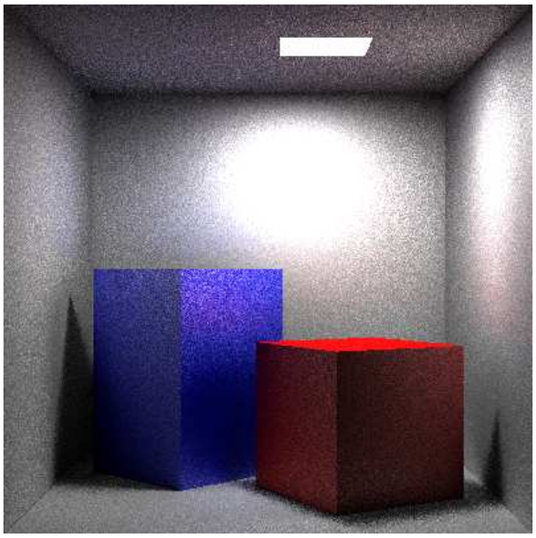

Fig. 6. BRDF based sampling with 100 samples per pixel

7 Conclusions

This paper presented a method that can fully exploit importance sampling strategies in the quasi-Monte Carlo solution of the rendering equation. The method transforms the low-discrepancy points according to the distribution of the estimated radiance generated from the knowledge gained in a photon-shooting step. This transformation requires some overhead, but can significantly reduce the number of required rays to attain a given error. Increasing the number of shot photons the estimation of the radiance function will be more accurate, but it needs more memory.

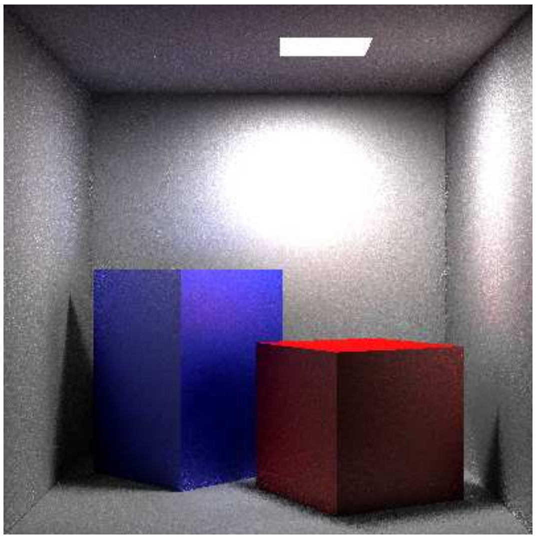

Fig. 7. Importance driven sampling with 100 samples per pixel

8 Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref.No.: F015884, T029135), the Austrian-Hungarian Action Fund (ref.No.: 29ö4 and 32öu9) and the Spanish-Hungarian Fund (ref.No.: E9).

References

[1] J. Arvo and D. Kirk. Particle transport and image synthesis. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 63–66, 1990.

[2] P. Dutre, E. Lafortune, and Y. D. Willems. Monte Carlo light tracing with direct computation of pixel intensities. In *Compugraphics '93*, pages 128–137, Alvor, 1993.

[3] H. W. Jensen. Importance driven path tracing using the photon maps. In *Rendering Techniques '95*, pages 326–335, 1995.

[4] A. Keller. A quasi-Monte Carlo algorithm for the global illumination in the radiosity setting. In H. Niederreiter and P. Shiue, editors, *Monte-Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 239–251. Springer, 1995.

[5] E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, 1993.

[6] E. Lafortune and Y. D. Willems. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques '96*, pages 11–19, 1996.

[7] B. Lange and B. Beyer. Rayvolution: An evolutionary ray tracing algorithm. In *Photorealistic Rendering Techniques*, pages 136–144, 1994.

[8] G. P. Lepage. An adaptive multidimensional integration program. Technical Report CLNS-80/447, Cornell University, 1980.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[10] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods.* SIAM, Pennsilvania, 1992.

[11] P. Shirley and C. Wang. Luminaire sampling in distribution ray tracing. *SIGGRAPH '93 Course Notes*, 1993.

[12] P. Shirley, C. Wang, and K. Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[13] L. Szirmay-Kalos. Global ray-bundle tracing. Technical Report TR-186-2-98-18, Institute of Computer Graphics, Vienna University of Technology, 1998. www.cg.tuwien.ac.at/.

[14] L. Szirmay-Kalos, B. Csébfalvi, and W. Purgathofer. Importance-driven quasi-Monte Carlo solution of the rendering equation. In *Winter School of Computer Graphics '98*, pages 377–386, Plzen, Czech Republic, 1998.

[15] L. Szirmay-Kalos, T. Fóris, L. Neumann, and B. Csébfalvi. An analysis to quasi-Monte Carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum (Eurographics'97)*, 16(3):271–281, 1997.

[16] E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.

[17] E. Veach and L. Guibas. Metropolis light transport. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 65–76, 1997.