Importance Driven Quasi-Random Walk Solution of the Rendering Equation

László Szirmay-Kalos, Balázs Csébfalvi, Werner Purgathofer
Department of Control Engineering and Information Technology, Technical University of Budapest,
Budapest, Műegyetem rkp. 11, H-1111, HUNGARY
szirmay@fsz.bme.hu

Abstract: This paper presents a new method that combines quasi-Monte Carlo quadrature with importance sampling to solve the general rendering equation efficiently. Since classical importance sampling has 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 explicitely, 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 and the patches are interconnected by links.

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 radiance reaching the camera from a given direction, i.e. through a given pixel, taking into account the optical properties of the surfaces and the lightsources in the virtual world. However, 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:

$$L(\vec{x},\omega) = L^e(\vec{x},\omega) + \int_{\Omega'} L(h(\vec{x},-\omega'),\omega')\cdot\cos\theta'\cdot f_r(\vec{x},\omega',\omega)\, d\omega', \quad (1)$$

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 ω', θ' is the angle between the surface normal and direction $-\omega'$, and $f_r(\vec{x},\omega',\omega)$ is the bi-directional reflection/refraction function (figure 1).

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

$$\int_{\Omega'} L(h(\vec{x},-\omega'),\omega')\cdot\cos\theta'\cdot f_r(\vec{x},\omega',\omega)\, d\omega' = \mathcal{T}L(\vec{x},\omega). \quad (2)$$

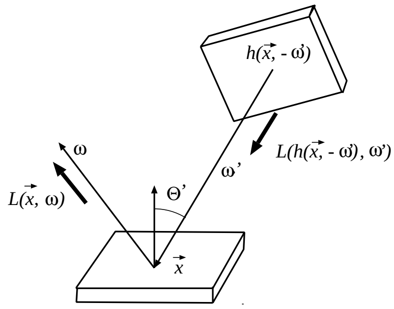

Figure 1: Geometry of the rendering equation

Thus the short form of the rendering equation is:

$$L = L^e + \mathcal{T}L. \quad (3)$$

There are two families of techniques that can be used to solve such an integral equation: finite element methods and random walk methods. Unlike finite-element methods, random-walk techniques need not require the surfaces to be decomposed into small patches but can work with the original geometry. This paper investigates the random-walk solution when the walk is not governed by random decisions but deterministic selection from appropriate uniform sequences.

In equation 3 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:

$$L = L^e + \mathcal{T}L = L^e + \mathcal{T}(L^e + \mathcal{T}L) =$$

$$L^e + \mathcal{T}L^e + \mathcal{T}^2 L = \sum_{i=0}^{\infty} \mathcal{T}^i L^e. \qquad (4)$$

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 study the structure of $\mathcal{T}^i L^e$, let us consider the case of $i = 2$:

$$\mathcal{T}^2 L^e = \mathcal{T}(\mathcal{T}L^e) =$$

$$\int\limits_{\Omega_1'} (\mathcal{T}L^e)(h(\vec{x}, -\omega_1'), \omega_1') \cdot \cos\theta_1' \cdot f_r(\vec{x}, \omega_1', \omega)\, d\omega_1' =$$

$$\int\limits_{\Omega_1'} \int\limits_{\Omega_2'} L^e(h(h(\vec{x}, -\omega_1'), -\omega_2'), \omega_2') \cdot \cos\theta_1' \cdot \cos\theta_2' \cdot$$

$$f_r(\vec{x}, \omega_1', \omega) \cdot f_r(h(\vec{x}, -\omega_1'), \omega_2', \omega_1',)\, d\omega_2' d\omega_1'. \qquad (5)$$

To evaluate the integrand at point $(\vec{x}, \omega, \omega_1', \omega_2')$, the following algorithm must be executed:

1. Surface point $\vec{y}_1 = h(\vec{x}, -\omega_1')$ — that is the point which is visible from $\vec{x}$ at direction $-\omega_1'$ — must be determined. This can be done by sending a ray from $\vec{x}$ into direction $-\omega_1'$ and indentifying the surface that is first intersected.

2. Surface point $\vec{y}_2 = h(h(\vec{x}, -\omega_1'), -\omega_2')$ — that is the point visible from $\vec{y}_1$ 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 $\vec{y}_2$ in the direction of ω_2' is ackquired and is multiplied with the cosine terms and the BRDFs of the two reflections.

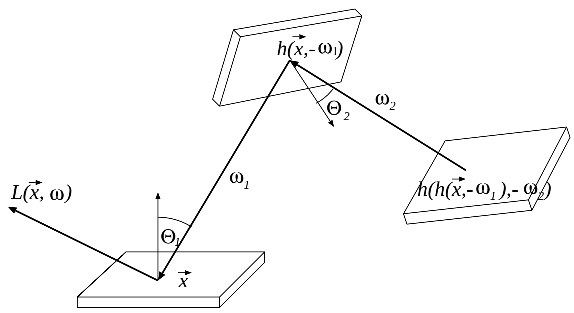

Figure 2: The integrand of $\mathcal{T}^2 L^e$ is a two-step walk

This algorithm can easily be generalized for arbitrary number of reflections. In order to compute the contribution of the nth reflection of the light to $L(\vec{x}, \omega)$, the following function should be integrated: A ray is emanated recursively from $\vec{x}$ at direction $-\omega_1'$ then from the found surface at $-\omega_2'$, etc. until $-\omega_n'$. 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.

These walks provide the value of the integrand at "point" $\omega_1', \omega_2', \ldots, \omega_n'$. The integral is estimated as the average of these values. Directions $\omega_1', \omega_2', \ldots, \omega_n'$ should be selected to minimize the error of the approximation.

1.1 Quasi-Monte Carlo Integration

For the sake of simplicity, assume that an s-dimensional funcion $f(\mathbf{z})$ 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:

$$\int\limits_{[0,1]^s} f(\mathbf{z})\, d\mathbf{z} \approx \frac{1}{M} \sum_{i=1}^{M} f(\mathbf{z}_i). \qquad (6)$$

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:

$$|\int\limits_{[0,1]^s} f(\mathbf{z})\, d\mathbf{z} - \frac{1}{M} \sum_{i=1}^{M} f(\mathbf{z}_i)| \le \mathcal{V}_{\mathrm{HK}} \cdot D^*(\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_M), \qquad (7)$$

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 [Nie92]. The star-discrepancy is defined by

$$D^*(\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_M) = \sup_A |\frac{m(A)}{M} - V(A)| \qquad (8)$$

where A is a s-dimensional subcube 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 subcube.

For carefully selected sample points, called *low-discrepancy sequences* [Nie92], the discrepancy and consequently the error can be in the order of $O(\log^{s-1} M/M)$, which is much better than the $O(1/\sqrt{M})$ probabilistic error bound of Monte-Carlo methods. Moreover, quasi-Monte Carlo methods guarantee this accuracy in a deterministic way, unlike Monte-Carlo methods where the error bound is also probabilistic.

1.2 Importance Sampling

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

$$\int_V f(\mathbf{z})\, d\mathbf{z} = \int_V \frac{f(\mathbf{z})}{p(\mathbf{z})} \cdot p(\mathbf{z})\, d\mathbf{z} =$$

$$E[\frac{f(\mathbf{z})}{p(\mathbf{z})}] \approx \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{f(\mathbf{z}_i)}{p(\mathbf{z}_i)} \pm \frac{\sigma}{\sqrt{M}}, \qquad (9)$$

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

$$\sigma^2 = E[(\frac{f(\mathbf{z})}{p(\mathbf{z})} - E[\frac{f(\mathbf{z})}{p(\mathbf{z})}])^2] =$$

$$\int_V (\frac{f(\mathbf{z})}{p(\mathbf{z})} - E[\frac{f(\mathbf{z})}{p(\mathbf{z})}])^2 \cdot p(\mathbf{z})\, d\mathbf{z}. \qquad (10)$$

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 distributions that are large where the integrand is large and small where the integrand is negligible.

2 Previous work

Monte-Carlo random walk methods generate finite 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.

In computer graphics the first Monte-Carlo random walk algorithm — called *distributed ray-tracing* — was proposed by Cook et al. [CPC84], which spawned to a set of variations, including *path tracing* [Kaj86], *light-tracing* [DLW93], *Monte-Carlo radiosity* [Shi91][Neu95][PM95], and two-pass methods which combine radiosity and ray-tracing [WCG87].

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*, emphasizing the starting point of the walk.

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

Thus 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 [DLW93]. *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 [VG94] only the last positions of the two walks are connected. Lafortune [LW93], 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 $L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$ 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 $\cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$. If we select the proportionality ratio as 1, then the probability density $\Pr\{\omega'\} = \cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$ will not be normalized, but integrates to a value less than 1. Since $\int_{\Omega'} \cos\theta' \cdot f_r(\vec{x}, \omega', \omega)\, d\omega'$ also expresses the probability that no reflection occures, this sub-critical probability density can be used as a termination criterion of the walk. Summarising, 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 very inhomogeneous. This happens if the scene contains small and bright lightsources.

The other direction, called as *direct lighting computation* [SWZ96], on the other hand, considers as important directions where the lightsources are visible. Thus their probability distributions are governed by the function $L^e(\omega')$. Direct lighting methods can become inefficient when the BRDFs are very inhomogeneous.

Realizing the complementary advantages of the two importance sampling techniques, Shirley et al. [SW93] used adaptive switching to select from the two alternatives, while Veach and Guibas [VG94] 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. The first paper, that used the product of BRDFs and lighting for importance sampling, stored the illumination information in a 5D octree [LW96].

Recently 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 [Kel95], 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 [SKFNC97]. However, the numerical evidence showed that quasi-Monte Carlo methods can slightly be better than Monte-Carlo techniques.

3 The new method

This paper aims at improving the previous techniques by proposing an algorithm that uses bi-directional 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 links are established between those patches that can illuminate each other.

In the second phase of the algorithm visibility ray-tracing walks (backward tracing) are executed for each pixel. The directions of the walk are generated from the links.

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.

3.1 Application of Importance Sampling in Quasi-Monte Carlo Integration

Classical Monte-Carlo integration places more samples where the integrand is large. The same basic idea is also sensible in quasi-Monte Carlo integration. However, for formal analysis, we have to find other approach since terms like probability density or variance cannot be applied in the deterministic quasi-Monte Carlo framework.

The alternative formulation is the integration using variable transformation. Suppose that a function f needs to be integrated in domain V and we have a monotonously increasing mapping T that maps this domain onto V'. The integral can also be evaluated in the new domain using the following formula:

$$\int_V f(\mathbf{z})\, d\mathbf{z} = \int_{V'} f(T^{-1}(\mathbf{y})) \left| \frac{\partial T^{-1}(\mathbf{y})}{\partial \mathbf{y}} \right| d\mathbf{y}, \quad (11)$$

where $\left|\frac{\partial T^{-1}(\mathbf{y})}{\partial \mathbf{y}}\right|$ is the Jacobi determinant of the inverse transformation.

If quasi-Monte Carlo integration is used, then domain V' is $[0,1]^s$. According to the Koksma-Hlawka inequality, the error of the quadrature depends on the variation of the function, which can be controlled by the approriate transformation of the integrand. In order to reduce the variation, the function should be flattened. In the ideal case when the integrand is constant, the variation is 0.

To make the transformed integrand constant, the Jacobi determinant should be inversely proportional to f. Since the intuitive meaning of the Jacobi determinant is the compression or expansion ratio of the two corresponding sub-cubes in V and V' respectively, this criterion states that if the sample points are uniformly distributed in V', then the transformation will concentrate them in regions where f is high.

When using this variable transformation method to implement importance sampling in quasi-Monte Carlo integration, two problems must be solved: the integrand f should be expressed and a mapping T must be found that has a Jacobi determinant which is inversely proportional to f.

In the finite Neumann series form of the radiance a 2-dimensional integral, a 4-dimensional integral,..., and a $2M$-dimensional integral should be evaluated that represent the 1-bounce, 2-bounce, ..., and M-bounce radiance transfers, respectively. It is obviously not worth evaluating these integrals separately, since they have different integrands, thus they would require different transformations to incorporate importance sampling. Thus rays generated, for instance, for the calculation of the 2-bounce transfer, could not be used for the estimation of the 1-bounce radiation.

The other alternative is the calculation of all integrals together as a single $2M$-dimensional integral using the following trivial identity:

$$\int_{V_x} f(x)\, dx + \int_{V_x}\int_{V_y} g(x,y)\, dx\, dy =$$

$$\int_{V_x}\int_{V_y} \frac{f(x)}{|V_y|} + g(x,y)\, dx\, dy \quad (12)$$

However, here the main difficulty is the determination and the inversion of an appropriate T mapping.

We shall show that such a T can be derived iteratively where only a 2-dimensional mapping is found at each step. For the sake of notational simplicity, assume that $M = 2$, thus only the first and second bounces are considered. The sum of the two integrals can be written as follows:

$$\mathcal{T}^1 L^e + \mathcal{T}^2 L^e = \int_{\Omega_1'} f(\omega_1')\, d\omega_1' + \int_{\Omega_1'}\int_{\Omega_2'} g(\omega_1', \omega_2')\, d\omega_1' d\omega_2' =$$

$$\int_{\Omega_1'} \left[f(\omega_1') + \int_{\Omega_2'} g(\omega_1', \omega_2')\, d\omega_2' \right] d\omega_1', \quad (13)$$

where

$$f(\omega_1') = L^e(h(\vec{x}, -\omega_1'), \omega_1') \cdot \cos\theta_1' \cdot f_r(\vec{x}, \omega_1', \omega)$$

and

$$g(\omega_1', \omega_2') =$$
$$L^e(h(h(\vec{x}, -\omega_1'), -\omega_2'), \omega_2') \cdot \cos\theta_1' \cdot \cos\theta_2'$$
$$\cdot f_r(\vec{x}, \omega_1', \omega) \cdot f_r(h(\vec{x}, -\omega_1'), \omega_2', \omega_1').$$

Assume that it is possible to find transformation $\mathbf{z}_1 = T_1(\omega_1')$ ($\mathbf{z}_1 \in [0,1]^2$) that makes the integrand of the outer integration a constant value, that is:

$$\int\limits_{\Omega_1'} \left[f(\omega_1') + \int\limits_{\Omega_2'} g(\omega_1', \omega_2')\, d\omega_2' \right] d\omega_1' =$$

$$\int\limits_{[0,1]^2} \left[f(T_1^{-1}(\mathbf{z}_1)) + \int\limits_{\Omega_2'} g(T_1^{-1}(\mathbf{z}_1), \omega_2')\, d\omega_2' \right] \left| \frac{\partial T_1^{-1}(\mathbf{z}_1)}{\partial \mathbf{z}_1} \right| d\mathbf{z}_1, \tag{14}$$

where:

$$\left[f(T_1^{-1}(\mathbf{z}_1)) + \int\limits_{\Omega_2'} g(T_1^{-1}(\mathbf{z}_1), \omega_2')\, d\omega_2' \right] \left| \frac{\partial T_1^{-1}(\mathbf{z}_1)}{\partial \mathbf{z}_1} \right| = c_1. \tag{15}$$

Furthermore, let us assume that we can also find a variable transformation $\mathbf{z}_2 = T_2(\omega_2')$ ($\mathbf{z}_2 \in [0,1]^2$) which makes the inner integrand constant for all possible $\mathbf{z}_1$, that is:

$$\int\limits_{\Omega_2'} g(T_1^{-1}(\mathbf{z}_1), \omega_2')\, d\omega_2' =$$

$$\int\limits_{[0,1]^2} g(T_1^{-1}(\mathbf{z}_1), T_2^{-1}(\mathbf{z}_2)) \left| \frac{\partial T_2^{-1}(\mathbf{z}_2)}{\partial \mathbf{z}_2} \right| d\mathbf{z}_2, \tag{16}$$

where:

$$g(T_1^{-1}(\mathbf{z}_1), T_2^{-1}(\mathbf{z}_2)) \left| \frac{\partial T_2^{-1}(\mathbf{z}_2)}{\partial \mathbf{z}_2} \right| = c_2. \tag{17}$$

Note that the variable transformation has been executed in two steps and in each step only two-dimensional variables have to be transformed. Now we show that this iterative method also guarantees that the integrand of the multi-dimensional quadrature is constant:

$$\int\limits_{[0,1]^2} \left[f(T_1^{-1}(\mathbf{z}_1)) + \int\limits_{[0,1]^2} g(T_1^{-1}(\mathbf{z}_1), T_2^{-1}(\mathbf{z}_2)) \left| \frac{\partial T_2^{-1}}{\partial \mathbf{z}_2} \right| d\mathbf{z}_2 \right] \cdot$$
$$\left| \frac{\partial T_1^{-1}}{\partial \mathbf{z}_1} \right| d\mathbf{z}_1 =$$
$$\int\limits_{[0,1]^2} \int\limits_{[0,1]^2} \left[f(T_1^{-1}(\mathbf{z}_1)) + g(T_1^{-1}(\mathbf{z}_1), T_2^{-1}(\mathbf{z}_2)) \left| \frac{\partial T_2^{-1}(\mathbf{z}_2)}{\partial \mathbf{z}_2} \right| \right] \cdot$$
$$\left| \frac{\partial T_1^{-1}(\mathbf{z}_1)}{\partial \mathbf{z}_1} \right| d\mathbf{z}_2\, d\mathbf{z}_1, \tag{18}$$

since the domain of $\mathbf{z}_2$ has unit size. The integrand of the multi-dimensional quadrature can be re-written using equation 15 and equation 17:

$$\left[f(T_1^{-1}(\mathbf{z}_1)) + g(T_1^{-1}(\mathbf{z}_1), T_2^{-1}(\mathbf{z}_2)) \left| \frac{\partial T_2^{-1}(\mathbf{z}_2)}{\partial \mathbf{z}_2} \right| \right] \left| \frac{\partial T_1^{-1}(\mathbf{z}_1)}{\partial \mathbf{z}_1} \right|$$

$$= \left[f(T_1^{-1}(\mathbf{z}_1)) + c_2 \right] \left| \frac{\partial T_1^{-1}(\mathbf{z}_1)}{\partial \mathbf{z}_1} \right| = c_1. \tag{19}$$

Thus, the variation of the multi-dimensional quadrature is also 0. Obviously, the same idea can also be used iteratively for considering more bounces.

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

1. 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 $L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$ 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. When a new point is identified, $L(h(\vec{x}, -\omega'), \omega') \cdot \cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$ is estimated and inverted again, then using 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 unknown radiance $L(h(\vec{x}, -\omega'), \omega')$ and the known, weighted BRDF which is $\cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$. In order to make this approach practically feasible, two problems need to be solved. We need effective methods to estimate radiance $L(h(\vec{x}, -\omega'), \omega')$ for a given $\vec{x}$ and to invert the product of the radiance and the BRDF.

3.2 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.

In order to store this information, the original surfaces are subdivided into patches that are connected by *links*. The intensity of the power between patches can be expressed by the intensity of the corresponding link. When the photon-shooting results in a surface-to-surface interaction, the links between the corresponding patches are increased by the power of the given interaction.

Having completed the photon-shooting step, the links store the information for each patch about where this neigborhood can be illuminated from. Assume that the domain of directions ω' is decomposed into finite number of "pixels" width the same area (for example, having the two polar coordinates of the given direction, the first one is uniformly quantized, and the other one is quantized according to the sine function). This array is called the *illumination map*.

The solid angle in which a linked patch is visible can be approximated with a rectangle painted in the illumination map. Assuming triangular patches, the nodes are projected to the directional sphere determining three pixels. The smallest rectangle which contains all the three pixels is painted in the illumination map with a color proportional to the intensity of the appropriate link. If the previous value of a pixel is higher then the new one than it keeps its value. This heuristic assumes that the patch corresponding to the highest intensity link overlays the others in the pixels which are common in the rectangles approximating the projected areas of different patches. This approach avoids the computationaly expensive visibility calculations, for example tracing of shadow rays.

If all linked patches are processed, then the illumination map is complete.

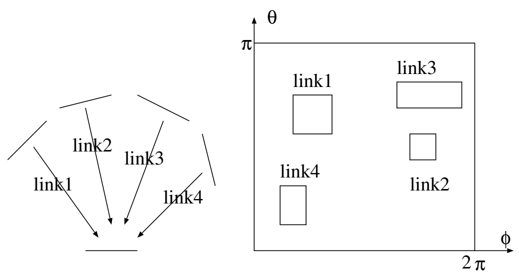

Figure 3: Correspondance between the links and the illumination map

3.3 Determination and inversion of the importance

Since the importance is proportional to the received radiance and the weighted BRDF, the illumination map is further processed, when each value in it is multiplied by $\cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$. This results in another array called the *importance map*. The inversion process should determine a mapping T that maps the importance map onto a unit size 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.

3.3.1 Mip-map based inversion

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

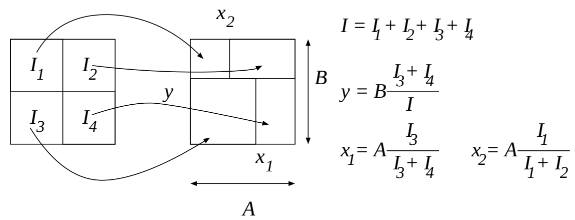

Figure 4: Mip-map organization of the importance map

An appropriate transformation that decomposes a rectangle of size $A \times B$ to 4 subrectangles having area proportional to the four colors is shown in figure 4. The inversion process — which generates a ϕ, θ point from a point in the unit 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 colors in the importance map.
3. The subrectangle that contains the given point if 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 consits 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, recursively applying the transformation of figure 4, 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.

3.3.2 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 determined from the center of the pixel adding some perturbation.

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

4 The algorithm

The photon-shooting step shoots photons of the same power. Suppose that the scene contains K number of lightsources having $L_l^e(\vec{x}, \omega)$ radiance power. 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 weighted BRDF $(\cos\theta' \cdot f_r(\vec{x}, \omega', \omega))$. 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 i, j coordinates of the Halton sequence:

for each lightsource l **do**
  $n_l = n \cdot \Phi_l / \Phi_T$
  Find starting point on lightsource 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
    link[source][destination] =
      link[source][destination] + Φ_T / n
    Generate reflection direction ω using
      $H(3 + i \cdot 2, 4 + i \cdot 2)$ transformed by
      by $\cos\theta' \cdot f_r(\vec{x}, \omega', \omega)$
    if not absorbed $\omega' = \omega$
    $i = i + 1$
  endwhile
endfor

This phase is followed by the view-dependent ray-tracing step that uses the links 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
    $color$ += $\text{RayTrace}(\vec{eye}, \omega, i, 1, 1)/n_{\text{sample}}$
    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 intersetion: $\vec{x}$, surface
  Generate Illumination map from the links $L[u][v]$ and
    from the projected areas
  Generate Importance map $I[u][v]$ multiplying $L[u][v]$
    and $\cos\theta'(u, v) \cdot f_r(\vec{x}, \omega'(u, v), \omega)$
  $I = \sum_u \sum_v I[u][v]$
  if $(atten < minatten)$
    Generate reflection direction ω' and find importance
      $I[u^*][v^*]$ using $H(2 \cdot level + 1, 2 \cdot level + 2)$
    $r = \cos\theta'(u^*, v^*) \cdot f_r(\vec{x}, \omega'(u^*, v^*), \omega) \cdot I / I[u^*][v^*]$
    return $L^e(\vec{x}, \omega) +$
      $A_{\text{pixel}} \cdot r \cdot \text{RayTrace}(\vec{x}, \omega', level + 1, atten \cdot r)$
  else return $L^e(\vec{x}, \omega) + A_{\text{pixel}} \cdot I$
end

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. As the level of the recursion increases, the importance map could get higher weights.

5 Implementation

The proposed algorithm was tested with 1M photons, where the resolution of the importance map was 16×16. In figure 5 the same scene is shown rendered with 10 and 100 samples per pixel. The first image demonstrates that the algorithm gives quite good approximation even taking small number of samples.

6 Conclusion

This paper presented a method that can fully exploit importance sampling strategies in the quasi-Monte Carlo solution of the rendering equation. The method

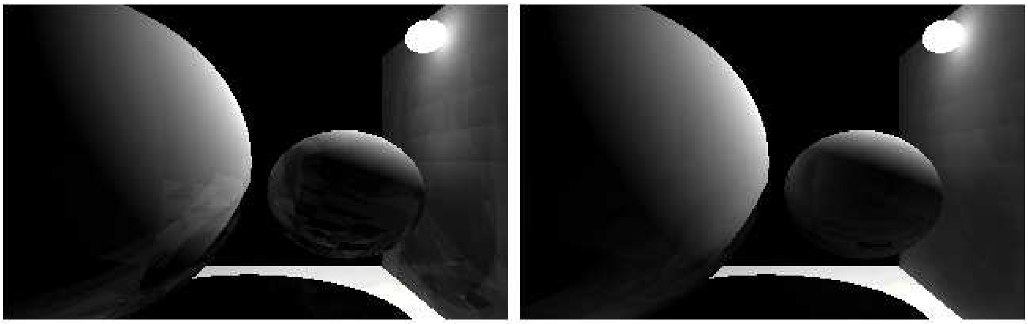

Figure 5: Simulation results (a: 10 samples/pixel, b: 100 samples/pixel)

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 capacity.

7 Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref.No.: F 015884) and the Austrian-Hungarian Action Fund (ref.No.: 29ö4 and 32öu9).

References

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of SIGGRAPH '84, Computer Graphics*, pages 137–145, 1984.

[DBW97] Ph. Dutre, Ph. Bekaert, and Y. Willems. Bidirectional radiosity. In *Rendering Techniques '97 (Proc. 8th EG Workshop on Rendering)*, pages 205–216, 1997.

[DLW93] 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, Portugal, 1993.

[DW96] Ph. Dutre and Y. Willems. Potential-driven monte carlo particle tracing for diffuse environments with adaptive probability functions. In *Rendering Techniques '96 (Proc. 7th EG Workshop on Rendering)*, pages 306–315, 1996.

[Kaj86] James T. Kajiya. The rendering equation. In *Proceedings of SIGGRAPH '86, Computer Graphics*, pages 143–150, 1986.

[Kel95] 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.

[LW93] E. Lafortune and Y.D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, Portugal, 1993.

[LW96] E. Lafortune and Y. Willems. A 5d tree to reduce the variance of monte carlo ray tracing. In *Rendering Techniques '96 (Proc. 7th EG Workshop on Rendering)*, pages 11–19, 1996.

[Neu95] L. Neumann. Monte carlo radiosity. *Computing*, 55:23–42, 1995.

[Nie92] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, Pennsilvania, 1992.

[PM95] S.N. Pattanik and S.P. Mudur. Adjoint equations and random walks for illumination computation. *ACM Transactions on Graphics*, 14(1):77–102, 1995.

[Shi91] P. Shirley. Discrepancy as a Quality Measure for Sampling Distributions. In *Eurographics '91*, pages 183–194, Amsterdam, North-Holland, 1991. Elsevier Science Publishers.

[SKFNC97] 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.

[SW93] P. Shirley and C. Wang. Luminaire sampling in distribution ray tracing. *SIGGRAPH '93 Course Notes*, 1993.

[SWZ96] P. Shirley, C. Wang, and K. Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[VG95] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *SIGGRAPH'95 Proceedings*, pages 419–428, 1995.

[WCG87] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *Proceedings of SIGGRAPH '87, Computer Graphics*, pages 311–324, 1987.