A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting

Alexander Keller

260/94

A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting

Alexander Keller

260/94

Universität Kaiserslautern
AG Numerische Algorithmen
Postfach 30 49
67653 Kaiserslautern

Nov. 1994

Herausgeber: AG Numerische Algorithmen
Leiter: Professor Dr. S. Heinrich

A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting

Alexander Keller

Fachbereich Informatik
AG Numerische Algorithmen
Universität Kaiserslautern
Postfach 3049, D-67653 Kaiserslautern
e-mail: keller@informatik.uni-kl.de

260/94

Abstract

The main problem in computer graphics is to solve the global illumination problem, which is given by a Fredholm integral equation of the second kind, called the radiance equation (REQ). In order to achieve realistic images, a very complex kernel of the integral equation, modelling all physical effects of light, must be considered. Due to this complexity Monte Carlo methods seem to be an appropriate approach to solve the REQ approximately. We show that replacing Monte Carlo by quasi-Monte Carlo in some steps of the algorithm results in a faster convergence.

1 Introduction

In computer graphics image generation [1] mainly consists of two steps: First the radiance equation (REQ) has to be solved to determine the global illumination. Then the solution of the REQ is sampled by simulating a lens in order to compute the radiance value for each pixel of the image matrix.

In our case the scene to be illuminated is composed of disjoint triangles A_k, $1 \leq k \leq K$, where K depends on the detail of modelling and is in the range of $10^3 \ldots 10^6$ for practical scenes. The union S of all triangles is the surface of the scene. We assume that the scene is finite and closed and that each triangle has a local coordinate system and is not degenerate. To solve the REQ means to know the radiance $L(x, \omega)$ in any location $x \in \cup_{k=1}^{K} A_k = S$ for any direction $\omega \in \Omega$, where Ω is the hemisphere in normal direction in x. The radiance L is measured in $[\frac{W}{m^2\, sr}]$ and as such is intensity per solid angle. Our eyes perceive something proportional to L. The radiance equation (REQ), a Fredholm integral equation of the second kind, is given by

$$L(x,\omega) = L_0(x,\omega) + \int_{\Omega'} L(h(x,\omega'), -\omega')\, f_r(-\omega', x, \omega)\, \cos\theta(x,\omega')\, d\omega' \tag{1}$$

where $L(x, \omega)$ is the radiance emerging from x in direction ω. L is the sum of the source radiance L_0 and the radiance from all over the hemishere Ω' (over surface location x) reflected at x into direction ω. $h(x, \omega')$ is the first point hit from location x in direction ω' and $\theta(x, \omega')$ is the angle between the surface normal in x and the direction ω'. The f_r term is a bidirectional reflection distribution function (BRDF) describing the surface properties in a surface location x for light coming from direction $-\omega'$ and being reflected into direction ω. In this way the BRDF characterizes color, gloss, etc. of the surface. Due to the Helmholtz principle we have

$$f_r(-\omega', x, \omega) = f_r(-\omega, x, \omega')$$

which means, that all light paths can be reversed. This is an important property and the basis of ray tracing. The algorithm which will be derived in the next section is designed for the so-called radiosity setting [2]. In this setting all surfaces are diffuse reflectors, i.e. we only allow constant BRDFs, describing a uniform reflection independent of direction:

$$f_r(x) = \frac{\rho_d(x)}{\pi}$$

The REQ is a convenient formulation of a radiative transfer problem. Usually the radiance L is a vector of components representing the color basis red, green and blue. So all equations would be replicated for each component. For the sake of simplicity we now restrict to only one wavelength.

We rewrite equation (1) as operator equation:

$$L = L_0 + T_f\, L \tag{2}$$

[1] For a profound introduction into computer graphics see [CW93], for a survey on Monte Carlo methods in computer graphics see [MP93].

[2] See the conclusion for elusion of this restriction.

The solution is accessible via the Neumann series. Since the operator norm is less than 1 in the diffuse environment, the series is converging and can be cut off at a certain level M with little loss of accuracy [3]:

$$L = \sum_{i=0}^{\infty} T_f^i L_0 \approx \sum_{i=0}^{M} T_f^i L_0 \tag{3}$$

Since the operator T_f is positive, the truncation of the series causes a certain underestimation.

The radiance value of one pixel P is the mean value integral over its area:

$$L_P = \frac{1}{|P|} \int_P L(h(Eye, \mathrm{dir}(Eye, x)), -\mathrm{dir}(Eye, x))\, dx \tag{4}$$

where Eye is the position of the observer's eye and $\mathrm{dir}(Eye, x)$ is the direction from the eye through location x on the screen into the scene.

2 Algorithm

The main idea of our algorithm is to calculate average local solutions of the REQ for the triangles A_k, as proposed in [HK94b], and then to reconstruct the image from these average informations. The average local solutions will be computed by quasi-Monte Carlo integration (see [Nie92]). The idea of applying low discrepancy particle methods to integral equations goes back to the work of [Hla62], [NW73], and [SP87].

In order to solve the REQ we insert (2) into itself and get

$$\begin{aligned} L &= L_0 + T_f\, L_0 + T_f\, T_f\, L \\ &= L_d + L_i \end{aligned} \tag{5}$$

where

$$\begin{array}{lll} L_d := & L_0 + T_f\, L_0 & \text{is the direct illumination and} \\ L_i := & T_f\, T_f\, L & \text{is the indirect illumination.} \end{array}$$

The direct illumination L_d is calculated separately, since there exist fast techniques for its computation. For the Monte Carlo integration, we have to integrate over all solid angles of triangles with $L_0 \neq 0$:

$$\begin{aligned} & L_d(x, \omega) \\ = \; & L_0(x, \omega) + \int_{\Omega'} L_0(h(x, \omega'), -\omega')\, f_r(-\omega', x, \omega)\, \cos\theta(x, \omega')\, d\omega' \\ = \; & L_0(x, \omega) + \sum_{k=1}^{LS} \int_{\Omega'_k} L_0(h(x, \omega'), -\omega')\, f_r(-\omega', x, \omega)\, \cos\theta(x, \omega')\, \chi_{A_k}(h(x, \omega'))\, d\omega' \\ \approx \; & L_0(x, \omega) + \sum_{k=1}^{LS} \frac{|\Omega'_k|}{N} \sum_{i=0}^{N-1} L_0(h(x, \omega'_i), -\omega'_i)\, f_r(-\omega'_i, x, \omega)\, \cos\theta(x, \omega'_i)\, \chi_{A_k}(h(x, \omega'_i)) \end{aligned} \tag{6}$$

[3]In practical diffuse environments $M \approx 10$ is sufficient.

where LS is the number of lightsources, Ω'_k is the solid angle of triangle A_k seen from location x without obstruction. The χ_{A_k} function is the visibility function:

$$\chi_{A_k}(h(x,\omega')) ::= \begin{cases} 1 & h(x,\omega') \in A_k \\ 0 & \text{else} \end{cases}$$

For the determination of the indirect contribution L_i, we first apply a functional Ψ_{A_k} to the solution of the REQ:

$$\langle L, \Psi_{A_k} \rangle = \int_S \int_\Omega L(x,\omega)\ \cos\theta(x,\omega)\ \chi_{A_k}(h(x,\omega))\ d\omega\ dx \tag{7}$$

The functional $\Psi_{A_k} = \cos\theta(x,\omega)\ \chi_{A_k}(h(x,\omega))$ is the "detector"-function for the total incoming power on surface A_k, which is given by the inner product above. So the mean diffuse radiance reflected by this triangle is

$$\overline{L_{A_k}}(x) = \frac{\rho_d(x)}{\pi\ |A_k|}\ \langle L, \Psi_{A_k} \rangle \tag{8}$$

Supposed we have an approximation for $\overline{L_{A_k}}(x)$, the approximation for the indirect illumination looks like:

$$\begin{aligned} L_i(x,\omega) &= (T_f\ T_f\ L)(x,\omega) \\ &\approx \int_{\Omega'} \left(\sum_{k=1}^{K} \overline{L_{A_k}}(h(x,\omega'))\ \chi_{A_k}(h(x,\omega')) \right) f_r(-\omega',x,\omega)\ \cos\theta(x,\omega')\ d\omega' \end{aligned} \tag{9}$$

To finally generate the image, we have to resample the approximation of L for each pixel. For one pixel we have

$$L_P \approx \frac{1}{OS} \sum_{i=0}^{OS-1} L(h(Eye, \mathrm{dir}(Eye, x_i)), -\mathrm{dir}(Eye, x_i)) \tag{10}$$

where OS is the oversampling rate per pixel, i.e. the number of rays shot from the eye through the pixel into the scene.

3 Quasi-Monte Carlo Integration

The quasi-Monte Carlo integration uses the same formula (11) as the (pseudo-) Monte Carlo method, but replaces the (pseudo-) random numbers by low discrepancy sequences.

$$\int_{[0,1]^s} f(u)\ du \approx \frac{1}{N} \sum_{i=0}^{N-1} f(u_i) \tag{11}$$

The (u_i) then are a deterministic pattern on $\bar{I}^s = [0,1]^s$ specifically designed for integration. That means, that quasi-random numbers may fail several statistical tests applied for pseudo-random numbers; they solely keep the property of uniform distribution. Since we can only use a finite number of samples, the pattern has a deviation from the uniform distribution measured as discrepancy. The discrepancy of a given (deterministic) point set $P_N = \{u_0, \ldots, u_{N-1}\}$ with regard to a family of subsets $\mathcal{F}$ is defined as

$$D(\mathcal{F}, P_N) = \sup_{A \in \mathcal{F}} \left| \lambda_s(A) - \frac{1}{N} \sum_{i=0}^{N-1} \chi_A(u_i) \right| , \tag{12}$$

where the supremum is taken over all subsets $A \subseteq \bar{I}^s$ of the family $\mathcal{F}$ and λ_s is the Lebesgue measure on $\bar{I}^s$. The discrepancy can be seen as maximum integration error for the characteristic functions of such a family of subsets.

For $\mathcal{J}^* = \{A = \prod_{i=1}^{s} [0, a_i] \mid 0 \leq a_i \leq 1\}$ we have the star-discrepancy

$$D^*(P_N) ::= D(\mathcal{J}^*, P_N) .$$

The inequality of Koksma and Hlawka

$$\left| \int_{\bar{I}^s} f(u)\, du - \frac{1}{N} \sum_{i=0}^{N-1} f(u_i) \right| \leq V(f) \cdot D^*(P_N) \tag{13}$$

is a separation of upper error bound into the star-discrepancy, depending only on the sampling pattern, and the total variation $V(f)$ of the integrand. This decomposition makes sense if the total variation $V(f)$ in the sense of Hardy and Krause (for further details see [Nie92]) can be bounded by a finite constant.

Since L, L_0 and f_r are not continuous, this estimation is not directly applicable. In [HK94a] we showed an upper error bound for the pixel oversampling (10), whereas an upper error bound for algorithms for L and $\overline{L_{A_k}}$ is an open problem in the setting of computer graphics.

3.1 Low Discrepancy Sequences

Let us now recall two standard types of low discrepancy sequences. First, we define Φ as the radical inverse function:

$$\Phi_b(i) = \sum_{j=0}^{\infty} a_j(i)\, b^{-j-1} \text{ when } i = \sum_{j=0}^{\infty} a_j(i)\, b^j \tag{14}$$

where the natural number $b > 1$ is the base and $i \in I\!N$. The values Φ_b always are in the unit interval $[0, 1)$. For illustration: $\Phi_2(i)$ simply is the binary representation of i mirrored at the decimal point [4].

The Halton and the Hammersley sequence are s-dimensional vectors built from radical inverse functions in relatively prime bases b_j, that is, we choose subsequent primes for the b_j. For $0 \leq i < N$ we have:

$$\begin{aligned} \text{Halton points: } u_i &= (\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i)) \\ \text{Hammersley points: } u_i &= (\frac{i}{N}, \Phi_{b_1}(i), \ldots, \Phi_{b_{s-1}}(i)) \end{aligned}$$

[4]For fast algorithms for computing Φ we refer to [Str93] or [HW64].

The discrepancies $D^*(P_N)$ for both sets have the following order of magnitude (see [Nie92]):

$$\begin{aligned} D^*_{Halton} &\in O\left(\frac{\log^s N}{N}\right) \\ D^*_{Hammersley} &\in O\left(\frac{\log^{s-1} N}{N}\right) . \end{aligned}$$

The Halton sequence is an incremental pattern, meaning that increasing the number of samples is possible without discarding the samples already drawn. In contrast to this the Hammersley sequence is not incremental since increasing the sampling rate results in discarding all samples computed so far. In consequence this pattern is not useful for adaptive sampling, although it has an asymptotically smaller discrepancy. No values of either sets need to be precomputed, since all values are directly accessible.

Using a permutation σ of the set $\{0, 1, \ldots, b-1\}$ for the generation of the radical inverse functions is called scrambling:

$$\Phi_b(i, \sigma) = \sum_{j=0}^{\infty} \sigma(a_j(i))\ b^{-j-1} \text{ when } i = \sum_{j=0}^{\infty} a_j(i)\ b^j$$

Scrambling is used for breaking the dependencies of the radical inverse function for large bases in high dimensions. Scrambling, which is useful in most applications of quasi-Monte Carlo integration, in computer graphics has no big effect. That fact arises from the complex scene geometry, where an ϵ-change of one ray direction can change a random walk path completely.

There is a variety of more low discrepancy sequences as the Faure-, the Sobol-, and further scrambled sequences. All these sequences differ a little bit in quality for the environment of computer graphics, but have about the same order of magnitude in discrepancy.

4 Implementation

The implementation is split into a preprocessing step for the evaluation of (7) and the rendering step for calculating the mean pixel radiances of (10).

We rewrite equation (7) by substituting the truncated Neumann series (3) for the solution L:

$$\begin{aligned} &\langle L, \Psi_{A_k} \rangle \\ &\approx \langle \sum_{j=0}^{M} T_f^j\ L_0, \Psi_{A_k} \rangle \\ &= \sum_{j=0}^{M} \langle T_f^j\ L_0, \Psi_{A_k} \rangle \\ &= \sum_{j=0}^{M} \int_S \int_\Omega (T_f^j\ L_0)(x, \omega)\ \cos\theta(x, \omega)\ \chi_{A_k}(h(x, \omega))\ d\omega\ dx \end{aligned}$$

$$= \sum_{j=0}^{M} \int_{\Omega^{j+1}} \int_{S_0} \prod_{l=1}^{j} f_r(-\omega_l, x_l, \omega_{l+1}) \prod_{l=1}^{j+1} \cos\theta(x_{l-1}, \omega_l)$$
$$L_0(x_0, \omega_1)\, \chi_{A_k}(x_{j+1})\, dx_0\, d\omega_1 \cdots d\omega_{j+1}$$

$$= \sum_{j=0}^{M} \int_{Q^{j+1}} \int_{S_0} \prod_{l=1}^{j} f_r(-\omega_l, x_l, \omega_{l+1}) \prod_{l=1}^{j+1} \frac{\sin 2\theta_l}{2}$$
$$L_0(x_0, \omega_1)\, \chi_{A_k}(x_{j+1})\, dx_0\, d\theta_1\, d\phi_1 \cdots d\theta_{j+1}\, d\phi_{j+1}$$

$$= \sum_{j=0}^{M} |S_0|\, \pi^{2j+2} \int_{I^{2j+4}} \prod_{l=1}^{j} f_r(-\omega_l, x_l, \omega_{l+1}) \prod_{l=1}^{j+1} \frac{\sin \pi u_{2l}}{2}$$
$$L_0(x_0, \omega_1)\, \chi_{A_k}(x_{j+1})\, du_0 \cdots du_{2j+3} \qquad (15)$$

where

$$\begin{aligned} x_l &= h(x_{l-1}, \omega_l) \text{ for } l > 0 \text{ and} \\ Q &= [0, \frac{\pi}{2}] \times [0, 2\pi] \end{aligned}$$

S_0 is the surface of the lightsources where $L_0 > 0$. For the Monte Carlo evaluation of (15) we use a $(2M+4)$-dimensional low discrepancy sequence (u_i) for all decisions of a random walk simulation. Using the first two components $(u_{i,0}, u_{i,1})$ we choose a starting point $x_0 \in S_0$ for the path on a light source. Then the particle is shot into direction $\omega_1 = (\theta_1, \phi_1) = (2\pi u_{i,2}, \frac{\pi}{2} u_{i,3})$. The triangle which is hit first in this direction receives the incoming power. Subsequently the particle is attenuated due to the BRDF in the hitpoint and traced into the next direction given by $(2\pi u_{i,4}, \frac{\pi}{2} u_{i,5})$. This procedure is extended to a path length of $M+1$ lines. By this we do the evaluation of (7) for all triangles A_k simultaneously by only one low discrepancy sequence.

As mentioned in section 3.1, for adaptive termination of the preprocessing step we have to apply infinite low discrepancy sequences like the Faure-, Halton-, or Sobol-sequence. The termination criterion is similar to [Pas94]. For two numbers N_1 and N_2 of iterated paths, we determine an error by

$$\triangle E(N_1, N_2) = \frac{1}{\sum_{k=1}^{K} L_{0,k}\, |A_k|} \sqrt{\frac{\sum_{k=1}^{K} d(\overline{L_{A_k}(N_1)}, \overline{L_{A_k}(N_2)})^2\, |A_k|}{\sum_{k=1}^{K} |A_k|}}$$

where the $\overline{L_{A_k}(N_i)}$ is the approximation of (7) by N_i samples and L_0 is the radiance of the different light sources. Further we select an interval $\triangle N$ for the measurements. The distance d is the Euclidean distance between the two color vectors $\overline{L_{A_k}(N_i)}$. The process is terminated if for a fixed T and the smallest $n \in \mathbb{N}$

$$\triangle E((n+t)\,\triangle N, (n+t+1)\,\triangle N) < \epsilon \text{ for } 0 \le t < T\ .$$

Since the error is weighted by the size of the triangles, after termination the bigger areas A_k are integrated more exactly than the smaller areas. This makes sense, because in the resampling step (9) the bigger areas are hit more often than the smaller ones.

The image generation is done by selecting positions in a pixel and sending rays from the eye through these positions into the scene. As shown in [HK94a], the Halton sequence is well suited for adaptive oversampling in this case.

Having determined the hitpoints in the scene, the radiance emerging from them into the direction of the eye has to be calculated by (5) for averaging (10). The direct light L_d is calculated by (6). It would be very expensive to calculate the contribution of all lights for all hitpoints of the oversampling process. So for the OS hitpoints, we calculate the light contribution of $\frac{\text{number of lights}}{OS}$ lights selected by a random permutation (which changes from pixel to pixel so as to avoid aliasing), so that alltogether every light is used exactly once for one pixel.

For each first hitpoint from the eye into the scene, we resample (7) by jittering rays over the hemisphere. From (9) we derive

$$\begin{aligned}
L_i(x,\omega) &\approx \int_{\Omega'} \left(\sum_{k=1}^{K} \overline{L_{A_k}}(h(x,\omega'))\, \chi_{A_k}(h(x,\omega')) \right) f_r(-\omega', x, \omega)\, \cos\theta(x,\omega')\, d\omega' \\
&= \int_{Q} \left(\sum_{k=1}^{K} \overline{L_{A_k}}(h(x,\omega'))\, \chi_{A_k}(h(x,\omega')) \right) f_r(-\omega', x, \omega)\, \frac{\sin 2\theta'}{2}\, d\theta'\, d\phi' \\
&= \int_{Q} \left(\sum_{k=1}^{K} \overline{L_{A_k}}(h(x,\omega'))\, \chi_{A_k}(h(x,\omega')) \right) f_r(-\omega', x, \omega)\, \frac{d \sin^2\theta'}{2}\, d\phi' \\
&\approx \frac{\pi}{SR} \sum_{i=0}^{SR-1} \left(\sum_{k=1}^{K} \overline{L_{A_k}}(h(x,\omega'))\, \chi_{A_k}(h(x,\omega'_i)) \right) f_r(-\omega'_i, x, \omega)
\end{aligned}$$

where $x = h(Eye, -\omega)$ is the point first hit by a ray from the Eye into direction $-\omega$. SR is the number of scattered rays used for the evaluation of the integral. For reasons of efficiency a kind of importance sampling is applied. The rays are not uniformly distributed over the hemisphere Ω, but distributed with respect to the $\sin^2\theta'$-term.

This sampling method applies to all BRDFs. In order to reduce the variance, it would be better to use a basis decomposition of the BRDF and to integrate over the basis vector separately (e.g. see [War92]). But since here we only allow the evaluation of diffuse, i.e. constant BRDFs no decomposition is necessary.

In addition to the algorithm explained above, we implemented two enhancements: First we included mirrors. A perfect mirror can be modelled by a δ-function. So whenever a ray hits a singular surface, it is reflected without changing its radiance. The second enhancement concentrates on the fact that the $h(x,\omega)$ operation is the most expensive operation in computer graphics. Therefore we implicitely increase the number of samples for the indirect light $\overline{L_{A_k}}$ of one pixel by using the samples of the eight neighbouring pixels. In order to assure a small error for this averaging, we have to guarantee a small deviation of normals. Therefore we only take those indirect light rays, which hit the same surfaces as considered in the current pixel. Since we exclusively used triangles to model our scenes, the deviation then is equal to zero. This kind of increment of samples acts as a low pass filter (since Monte Carlo integration simply is averaging) and so smoothes the indirect light contribution. Note that this method does not smear sharp contours, because it is not an image space method.

This method acts conveniently to the observer's eye: If we see areas which extend over big areas of the image, we tend to perceive high levels of noise. Exactly in this case the averaging smoothes this effect. If on the other hand we have many small objects in one pixel, the eye cannot distinguish and such does not perceive the high variance.

4.1 Measurements

Due to the complexity of the kernel of the REQ, for realistic scene description no analytic solution is accessible to serve as benchmark for the various rendering methods. So there are two ways left to test the performance of illumination algorithms. The first is to compute a solution with a very high number of samples and to take this solution as reference for calculations with less samples (see [HK94b]).

The second method is to simplify the kernel in such a way that an analytical solution becomes accessible. Be $\rho_d = \frac{1}{2}$, $f_r = \frac{\rho_d}{\pi}$, and $L_0 = \frac{1}{4}$ for all triangles A_k. Then the solution

$$\begin{aligned} L &= L_0 + \int_\Omega \frac{\rho_d}{\pi} L \cos\theta(\omega)\, d\omega \\ &= L_0 + T_f L = \sum_{i=0}^{\infty} T_f^i L_0 \\ &= \sum_{i=0}^{\infty} \left(\frac{1}{2\pi}\right)^i \frac{1}{4} T^i = \frac{1}{4} \sum_{i=0}^{\infty} \frac{1}{2^i} = \frac{1}{2} \end{aligned}$$

where

$$T = \int_\Omega \cos\theta\, d\omega = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \cos\theta(\omega)\, \sin\theta\, d\theta\, d\phi = \pi$$

simply is the projection of the unit-hemisphere onto the plane. By simplifying the kernel this way, only the complexity of the construction of the scene remains. This analytic solution can be used for finding obvious errors in the simulation. But more importantly it can serve as a benchmark test for illumination algorithms by calculating the distance to the analytic solution.

For the experiments we chose four different scenes. All scenes were modelled in meters, that is in real measure. The scenes are an empty cube ($K = 12$ triangles), a living room ($K = 3604$ triangles), an office ($K = 276$ triangles) and the computer graphics teapot (*Utah Teapot*) in a box ($K = 1572$ triangles). Images of the last three scenes can be seen in figure 1.

The algorithm described in section 2 will always have a small systematic error due to the truncation of the Neumann series. In the measurements we used $M = 29$ for the experiments with the analytic solution in order to keep the truncation error very small and $M = 6$ for the realistic experiments.

For the analytic solution experiments (all surfaces were grey with $f_r = \frac{1}{2\,\pi}$ and emitted white light with $L_0 = \frac{1}{4}$) we calculated the mean square deviation $\Delta_a(N)$ and the weighted mean square deviation $\Delta_{aw}(N)$:

$$\Delta_a(N) = \sqrt{\frac{\sum_{k=1}^{K} (\overline{L_{A_k}(N)} - \frac{1}{2})^2}{K}}$$

$$\Delta_{aw}(N) = \sqrt{\frac{\sum_{k=1}^{K}(\overline{L_{A_k}(N)} - \frac{1}{2})^2|A_k|}{K\sum_{k=1}^{K}|A_k|}}$$

The comparison of different sampling patterns for various numbers of samples is illustrated in tables 2 and 3. It can be observed that the low discrepancy sequences acquire the same level of error faster than the (pseudo-) random sequences. Note that inside the cube there are not any obstructions, and so the algorithm converges very fast due to the simple, symmetric geometry.

For the realistic setting (i.e. real surface textures and light sources, except for the cube, for which we still used the analytic setting), table 1 shows the behaviour of adaptive termination as described above. For a given accuracy ϵ of the preprocessing step, the number N of samples needed for termination ($\Delta N = 1000$, $T = 2$) is printed. From these tables it can be seen that the complexity of construction of the scene nearly replaces scrambling of the low discrepancy sequences. It also can be seen that the low discrepancy sequences are superior to the (pseudo-) random sequences, i.e. they terminate faster.

5 Conclusion and Further Work

We proposed an algorithm to approximately solve the global illumination problem for scenes consisting of diffuse and pure specular objects. The quasi-random, i.e. deterministic preprocessing step is applicable to any BRDF. In order to make the resampling step apply to not only diffuse BRDF and especially to gather caustics, the incoming radiance in equation (7) has to be stored for any incoming direction (for example by spherical harmonics).

By our experiments we proved that the quasi-random approach for the preprocessing step is slightly superior to the pseudo-random algorithm.

6 Acknowledgement

The author would like to thank Stefan Heinrich for helpful discussions and suggestions.

References

[CW93] M.Cohen, J.Wallace: Radiosity and Realistic Image Synthesis, Academic Press Professional, Cambridge 1993.

[HW64] J.H.Halton, G.Weller: Algorithm 247: Radical-inverse quasi-random point sequence [G5], Comm. ACM, No.12, 7(1964), pp. 701-702.

[HK94a] S.Heinrich, A.Keller: Quasi-Monte Carlo methods in computer graphics, Part I: The QMC-Buffer, Technical Report 242/94, University of Kaiserslautern, 1994.

[HK94b] S.Heinrich, A.Keller: Quasi-Monte Carlo methods in computer graphics, Part II: The Radiance Equation, Technical Report 243/94, University of Kaiserslautern, 1994.

[Hla62] E.Hlawka: Lösung von Integralgleichungen mittels zahlentheoretischer Methoden I, Sitzungsber., Abt. II, Österr. Akad. Wiss., Math.-Naturwiss. Kl., 171(1962), pp. 103-123.

[MP93] S.P.Mudur, S.N.Pattanaik: Monte Carlo methods for computer graphics, in State of the Art Reports, Eurographics, 1993.

[NW73] H.Neunzert, J.Wick: Die Theorie der asymptotischen Verteilung und die numerische Lösung von Integrodifferentialgleichungen, Numer. Math., 21(1973), pp. 234-243.

[Nie92] H.Niederreiter: Random Number Generation and Quasi-Monte Carlo Methods, SIAM Philadelphia, Pennsylvania 1992.

[Pas94] S.Paskov: Termination Criteria for Linear Problems, Technical Report, to appear, Columbia University, 1994.

[SP87] P.K.Sarkar, M.A.Prasad: A comparative study of pseudo and quasi random sequences for the solution of integral equations, J. Comp. Physics, 68(1987), pp. 66-88.

[Str93] J.Struckmeier: Fast generation of low-discrepancy sequences, Berichte der Arbeitsgruppe Technomathematik, Nr.93, Fachbereich Mathematik, Universität Kaiserslautern.

[War92] G.Ward: Measuring and Modeling Anisotropic Reflection, Computer Graphics, No.2, 26(1992), pp. 265-272.

Sequence	N per scene			
	cube	living room	office	teapot
$\epsilon = 10^{-3}$				
Lin. Congr.	42000	36000	48000	31000
Inversive	35000	37000	37000	36000
Halton	33000	36000	28000	29000
Faure	32000	29000	30000	30000
Sobol	30000	35000	30000	24000
$\epsilon = 3.3 \cdot 10^{-4}$				
Lin. Congr.	100000	99000	149000	102000
Inversive	101000	115000	126000	106000
Halton	92000	100000	72000	85000
Faure	99000	95000	72000	77000
Sobol	95000	87000	78000	77000
$\epsilon = 10^{-4}$				
Lin. Congr.	330000	343000	439000	334000
Inversive	337000	314000	397000	316000
Halton	266000	305000	235000	238000
Faure	276000	283000	276000	219000
Sobol	310000	298000	253000	228000

Table 1: Samples needed for a termination by ϵ

Samples	Sequence				
N	Lin. Congr.	Inversive	Halton	Faure	Sobol
empty cube					
10^3	0.0647699	0.0469468	0.0459244	0.0375936	0.0458288
10^4	0.0138841	0.0150312	0.015322	0.0101492	0.0154578
10^5	0.00416856	0.00426497	0.00347108	0.00388516	0.0139228
living room					
10^3	5.68352	3.89019	2.0911	1.46717	5.49831
10^4	1.87948	1.5128	2.15754	1.94902	1.51938
10^5	0.64136	0.550875	0.524932	0.727295	0.573517
office					
10^3	0.46748	0.456453	0.399271	0.280669	0.431063
10^4	0.259767	0.202392	0.331709	0.475231	0.234959
10^5	0.136946	0.127394	0.136938	0.136222	0.14049
teapot					
10^3	3.55052	24.4236	5.28716	3.58575	4.63836
10^4	1.36052	2.67153	0.962619	1.01024	0.920303
10^5	0.460942	0.438209	0.434956	0.412762	0.422177

Table 2: Mean square deviation from analytical solution

Samples	Sequence				
N	Lin. Congr.	Inversive	Halton	Faure	Sobol
empty cube					
10^3	0.00589147	0.00485275	0.00514532	0.00398542	0.00500996
10^4	0.00171686	0.00181801	0.00147316	0.00128062	0.00184561
10^5	0.000478878	0.000538219	0.000402798	0.000395118	0.00152823
living room					
10^3	0.0056218	0.00591197	0.00478811	0.00455163	0.00582923
10^4	0.00325591	0.00320881	0.00322973	0.00330551	0.00322495
10^5	0.0028332	0.00282448	0.00282852	0.00282892	0.00284094
office					
10^3	0.00861589	0.00845805	0.00825801	0.0070116	0.00775931
10^4	0.00672887	0.00668762	0.00670869	0.00680193	0.00695615
10^5	0.00661765	0.00654544	0.00655064	0.00657714	0.00685
teapot					
10^3	0.00624067	0.0116751	0.00809856	0.00610271	0.00580429
10^4	0.00183252	0.00199039	0.00156816	0.00158062	0.00159681
10^5	0.000581887	0.000552315	0.000543979	0.000565605	0.000613286

Table 3: Weighted mean square deviation from analytical solution

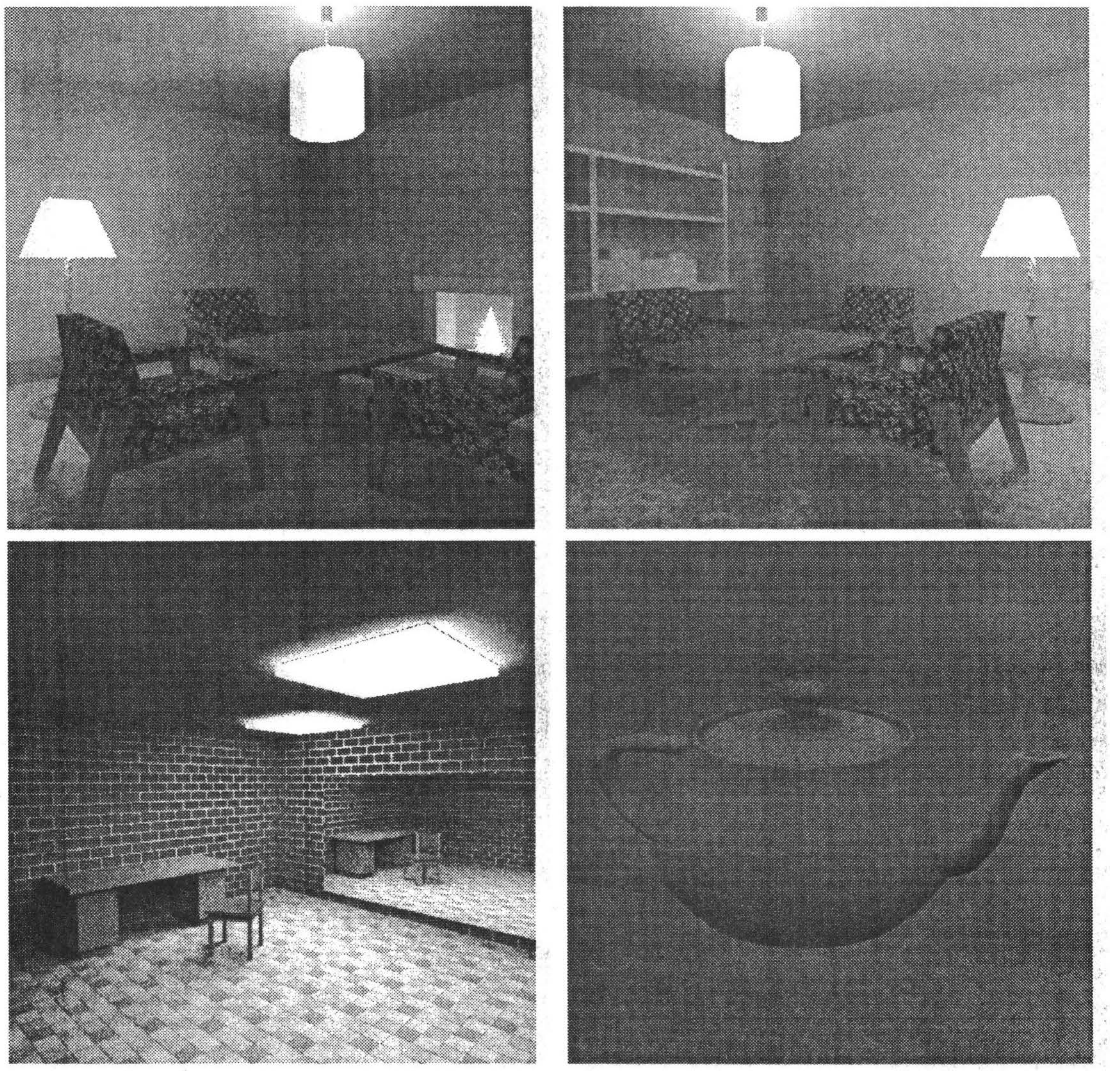

Figure 1: Images of the living room, the office and the teapot