Global Ray-bundle Tracing with Infinite Number of Rays

László Szirmay-Kalos

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

Werner Purgathofer

Institute of Computer Graphics, Vienna University of Technology

Abstract

The paper presents a combined finite element and quasi-random walk method to solve the general rendering equation. Applying finite element techniques, the surfaces are decomposed into planar patches that are assumed to have position independent, but not direction independent (that is non-diffuse) radiance. The direction dependent radiance function is then computed by quasi-random walk. Since quasi-Monte Carlo quadrature is applied here to an integrand of finite variation, this method can take advantage of the superior convergence of quasi-Monte Carlo integration.

1 Introduction

Global illumination algorithms aim at obtaining the power detected by a collection of measuring devices. The measurement process is characterized by an operator $\mathcal{M}$ which is applied to the radiance function. The radiance function, on the other hand, can be obtained by solving the *rendering equation* [11] that has the following form:

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

where $L(\vec{x},\omega)$ and $L^e(\vec{x},\omega)$ are the radiance and emission of the surface in point $\vec{x}$ at direction ω, Ω is the directional sphere, $h(\vec{x},\omega')$ is the visibility function defining the point that is visible from point $\vec{x}$ at direction ω', $f_r(\omega',\vec{x},\omega)$ is the bi-directional reflection/refraction function, θ' is the angle between the

Preprint submitted to Elsevier Preprint

surface normal and direction $-\omega'$, and $\cos^* \theta' = \cos \theta'$ if $\cos \theta' \geq 0$ and zero otherwise. Denoting the integral operator by $\mathcal{T}$, the rendering equation can also be written as $L = L^e + \mathcal{T}L$.

Since the rendering equation contains the unknown radiance function both inside and outside the integral, in order to express the solution, this coupling should be resolved. The possible solution techniques fall into one of the following three categories: inversion, expansion or iteration.

Inversion groups the terms that contain the unknown function on the same side of the equation and applies formally an inversion operation:

$$(1 - \mathcal{T})L = L^e \quad \Longrightarrow \quad L = (1 - \mathcal{T})^{-1} L^e. \tag{2}$$

Thus the measured power is

$$\mathcal{M}L = \mathcal{M}(1 - \mathcal{T})^{-1} L^e. \tag{3}$$

However, since $\mathcal{T}$ is infinite dimensional, it cannot be inverted in closed form. Thus it should be approximated by a finite dimensional mapping, that is usually given as a matrix. The inversion is then the solution of the resulting linear equation.

Expansion substitutes function L in the right side by the complete right side (which equals to L) recursively. If the integral operator is a contraction, this provides the solution in the form of an infinite Neumann series:

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

Thus the measured power is

$$\mathcal{M}L = \sum_{i=0}^{\infty} \mathcal{M}\mathcal{T}^i L^e. \tag{5}$$

The main problem of expansion techniques is that they require the evaluation of very high dimensional integrals that appear as terms in the infinite series. In order to avoid dimensional explosion — i.e. evaluating a D-dimensional integral with $O(M^D)$ time complexity — Monte-Carlo [31] or quasi-Monte Carlo [19] techniques should be used. Since a single point of the integrands corresponds to a light path from the eye to a lightsource, and Monte-Carlo methods generate the reflection directions randomly, expansion methods are also called *random walk methods*.

In computer graphics the first Monte-Carlo random walk algorithm — called *distributed ray-tracing* — was proposed by Cook et al. [7], which spawned to a set of variations, including *path tracing* [11], *light-tracing* [8], *bi-directional path tracing* [14,36], *Monte-Carlo radiosity* [27,16,20], and *two-pass methods* which combine radiosity and ray-tracing [26,40,38].

To reduce the variance of the Monte-Carlo integration, most of these methods incorporate some form of *importance sampling* [31]. The importance can be based on the local BRDFs [8,14], on the direct illumination [28], on both [35,15], or can even be explored adaptively [37].

Expansion techniques generate random walks independently. It can be an advantage, since these algorithms can be parallelized easily. However, it also means that these methods "forget" the previous history of walks and cannot reuse the visibility information gathered in previous walks, thus they are not as fast as they could be.

Some sort of reusing the visibility information for many walks happens, for example, in the *multi-path methods*. They are essentially random walk methods, but in their single step many random walks are advanced parallely. Sbert [21,24,23] proposed a complete family of multi-path methods, that are based on random global lines, which is the basic "engine" to advance the walks.

Iterational techniques realize that the solution of integral equation (1) is the fixed point of the following iterational scheme

$$L_n = L^e + \mathcal{T} L_{n-1}, \tag{6}$$

thus if operator $\mathcal{T}$ is a contraction, then this scheme will converge to the solution from any initial function L_0. The measured power can be obtained as a limiting value

$$\mathcal{M}L = \lim_{n \to \infty} \mathcal{M}L_n. \tag{7}$$

In order to store the approximating functions L_n, usually finite-element techniques are applied, as for example, in diffuse radiosity [6], or in non-diffuse radiosity using partitioned hemispheres [10], directional distributions [29] or illumination networks [3].

Iteration has two critical problems. On the one hand, since the domain of L_n is 4 dimensional, an accurate finite-element approximation usually requires very many basis functions, which, in turn, need a lot of storage space. Although, *hierarchical methods* [9,1], *wavelet* or *multiresolution methods* [5,25] and *clustering* [30,4,32] can help, the memory requirements are still prohibitive for complex scenes.

Realizing that the main problem of deterministic approaches is the approximation of the transport operator and the application of this approximation, a crucial part of designing such an the algorithm is finding an accurate and "small" approximation. In the *well-distributed ray-sets* [17,2] a discrete approximation of the transport operator is generated adaptively, taking into account the important patches and directions. The adaptation strategy is to refine the discrete approximation when the iteration with the coarse approximation is already stabilized.

On the other hand, when finite element techniques are applied, operator $\mathcal{T}$ is only approximated, which introduces some non-negligible error in each step. If the contraction ratio of the operator is λ, then the total accumulated error will be approximately $1/(1-\lambda)$ times the error of a single step [33], which can be unacceptable for highly reflective scenes. For highly reflective scenes, the iteration is slow and the result is inaccurate if the approximation of the operator is not very precise. Very accurate approximations of the transport operator, however, require a lot of computation time and storage space.

2 Global ray-bundle tracing method

This section presents a new algorithm that decomposes the scene into a finite mesh. The light transfer between the patches is described by an integral equation consisting of surface and directional integrals. First, the surface integral is handled by finite element methods, then the resulting integral equation, which contains only directional integration, is solved by Monte-Carlo or quasi-Monte Carlo walks.

Walk methods proposed so far use individual ray-paths as samples of the integrand of the rendering equation. However, ray-shooting may waste a lot of computation by ignoring all the intersections but the one closest to the eye. Thus it seems worth using a set of *global directions* [21,16,23,22] for the complete scene instead of solving the visibility problem independently for different points $\vec{x}$. Moreover, ray-shooting is a simple but by no means the most effective visibility algorithm since it is unable to take advantage of image or object coherence. Other methods based on the exploitation of image coherence, such as the z-buffer, painter's, Warnock's, etc. algorithms can be considered as handling a bundle of parallel rays and solving the visibility problem for all of them simultaneously. Continuous (also called object-precision) methods can even determine the visibility problem independently of the resolution, which corresponds to tracing infinitely many parallel rays simultaneously [34].

These visibility algorithms assume that the surfaces are decomposed into planar patches, thus the proposed method also uses this assumption. On the

other hand, the patch decomposition also serves as the construction of the finite-element structure.

Recall that even single-ray techniques use recursive ray-tracing to simulate multiple interreflections. Thus ray-bundles should also be traced several times in different directions to model multiple interreflections. This tracing composes a walk using a ray-bundle in each step.

2.1 Computation of global ray-bundle walks

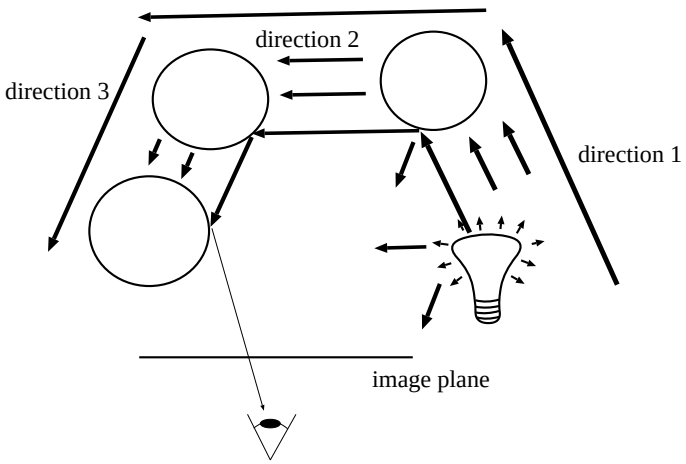

Fig. 1. A path of ray-bundles

The algorithm takes samples of the global walks and uses them in the quadrature. A single walk starts by selecting a direction either randomly or quasi-randomly, and the emission transfer of all patches is calculated into this direction (figure 1). Then a new direction is found, and the emission is transferred and the irradiance generated by the previous transfer is reflected from all patches into this new direction. The algorithm keeps doing this for a few times depending on how many bounces should be considered, then the emission is sent and the irradiance caused by the last transfer is reflected towards the eye. Averaging these contributions results in the final image. When the radiance reflection is calculated from the previous direction to the current direction or to the direction of the eye, the radiance is attenuated by the BRDF of the corresponding surface element.

Concerning the memory requirements of the method, each patch holds the incoming radiance of the last step of the walk and the accumulated radiance towards the eye. Since the selected directions are the same for all surfaces, they must be stored only once. Consequently the memory requirement is comparable to that of the diffuse radiosity algorithms although it is also capable to handle specular reflections.

In order to make this method work, two problems must be solved. The directions should be selected in a way that all the possible light-paths are covered, and the integral quadrature should be accurately approximated. The application of random or low-discrepancy series on the directional sphere is proposed to solve this problem.

Secondly, efficient algorithms are needed that can compute the radiance transfer of all patches in a single direction, for which generalizations of discrete and continuous visibility algorithms are applied.

2.2 Reformulation of the rendering equation using a finite-element method

According to the concept of finite-elements, the radiance, emission and the BRDF of patch i are assumed to be constant and are denoted by $L_i(\omega)$, $L_i^e(\omega)$ and $\tilde{f}_i(\omega', \omega)$, respectively. It means that the radiance function is approximated in the following way:

$$L(\vec{x}, \omega) \approx L^{(n)}(\vec{x}, \omega) = \sum_{j=1}^{n} L_j(\omega) \cdot b_j(\vec{x}) \tag{8}$$

where basis function $b_j(\vec{x})$ can, for example, be defined to be 1 if $\vec{x}$ is on patch j and 0 otherwise.

Using these assumptions, integrating the rendering equation on patch i we get:

$$L_i(\omega) \cdot A_i = L_i^e(\omega) \cdot A_i + \int_{\Omega} \int_{A_i} L(h(\vec{x}, -\omega'), \omega') \cdot \cos^* \theta' \cdot \tilde{f}_i(\omega', \omega) \; d\vec{x} \; d\omega'.$$

Taking into account that the integrand of the inner surface integral is piecewise constant, it can also be presented in closed form:

$$\int_{A_i} L(h(\vec{x}, -\omega'), \omega') \cdot \cos^* \theta' \cdot \tilde{f}_i(\omega', \omega) \; d\vec{x} = \sum_{j=1}^{n} \tilde{f}_i(\omega', \omega) \cdot A(i, j, \omega') \cdot L_j(\omega'),$$

where $A(i, j, \omega')$ expresses the projected area of patch j that is visible from patch i. In the unoccluded case this is the intersection of the projections of patch i and patch j onto a plane perpendicular to ω'. If occlusion occurs, the projected areas of other patches that are in between patch i and patch j should be subtracted as shown in figure 2.

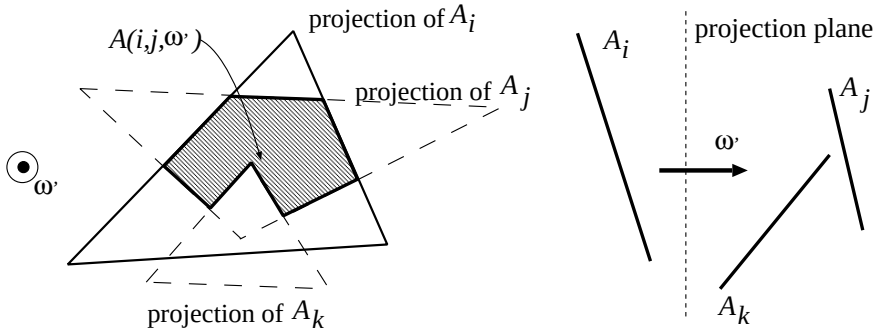

Fig. 2. Interpretation of $A(i, j, \omega')$

Using this, the original rendering equation is:

$$L_i(\omega) \cdot A_i = L_i^e(\omega) \cdot A_i + \int_\Omega \sum_{j=1}^{n} \tilde{f}_i(\omega', \omega) \cdot A(i, j, \omega') \cdot L_j(\omega') \; d\omega'.$$

Dividing both sides by A_i and converting the equation into matrix form, we obtain:

$$\mathbf{L}(\omega) = \mathbf{L}^e(\omega) + \int_\Omega \mathbf{F}(\omega', \omega) \cdot \mathbf{A}(\omega') \cdot \mathbf{L}(\omega') \; d\omega' = \mathbf{L}^e(\omega) + \mathcal{T}_F \mathbf{L}(\omega). \quad (9)$$

where $\mathbf{L}(\omega)|_i = L_i(\omega)$ is the vector of radiance values, and

$$\mathbf{A}(\omega')|_{ij} = \frac{A(i, j, \omega')}{A_i}$$

is the matrix of visible projected areas. This matrix is called the *geometry matrix.*

Note that equation (9) is highly intuitive as well. The radiance of a patch is the sum of the emission and the reflection of all irradiances into this direction. The role of the patch-direction-patch "form-factors" is played by $A(i, j, \omega')/A_i$.

2.3 Random or quasi-random walk

To obtain the solution, equation (9) is expanded into a Neumann series

$$\begin{aligned} \mathbf{L}(\omega) &= \sum_{i=0}^{\infty} \mathcal{T}_F^i \mathbf{L}^e(\omega) = \\ &= \mathbf{L}^e(\omega) + \int_\Omega \mathbf{F}(\omega', \omega) \cdot \mathbf{A}(\omega') \cdot \mathbf{L}^e(\omega_1') \; d\omega_1' \end{aligned}$$

$$+\int_{\Omega}\int_{\Omega}\mathbf{F}(\omega_1',\omega)\cdot\mathbf{A}(\omega_1')\cdot\mathbf{F}(\omega_2',\omega_1')\cdot\mathbf{A}(\omega_2')\cdot\mathbf{L}^e(\omega_2')\;d\omega_2'd\omega_1'$$
$$+\ldots \tag{10}$$

In practice the infinite sum of the Neumann series is always approximated by a finite sum. The number of required terms is determined by the overall reflectivity of the scene. Let us denote the maximum number of calculated bounces by D. The truncation of the Neumann series introduces a bias in the estimation, which can be tolerated if D is high enough.

In order to simplify the notations, we introduce the *d-bounce irradiance* $\mathbf{J}_d$ for $d = 1, 2, \ldots$ as follows:

$$\mathbf{J}_0 = \mathbf{A}(\omega_D')\cdot\mathbf{L}^e(\omega_D'),$$
$$\mathbf{J}_d = 4\pi\cdot\mathbf{A}(\omega_{D-d}')\cdot\mathbf{F}(\omega_{D-d+1}',\omega_{D-d}')\cdot\mathbf{J}_{d-1}$$

where $\mathbf{J}_d$ is a $d+1$ dimensional function of directions $(\omega_{D-d}', \omega_{D-d+1}'\ldots,\omega_D')$. The d-bounce irradiance represents the irradiance arriving at each patch, that is emitted from a patch and is bounced exactly d times.

Similarly, we can define the *max d-bounce irradiance* $\mathbf{I}_d$ for $d = 1, 2, \ldots$ as follows:

$$\mathbf{I}_0 = \mathbf{A}(\omega_D')\cdot\mathbf{L}^e(\omega_D'),$$
$$\mathbf{I}_d = \mathbf{A}(\omega_{D-d}')\cdot(\mathbf{L}^e(\omega_{D-d}') + 4\pi\cdot\mathbf{F}(\omega_{D-d+1}',\omega_{D-d}')\cdot\mathbf{I}_{d-1})$$

where $\mathbf{I}_d$ is a $d+1$ dimensional function of directions $(\omega_{D-d}', \omega_{D-d+1}'\ldots,\omega_D')$. The max d-bounce irradiance represents the irradiance arriving at each patch after completing a path of length d following the given directions, and gathering and then reflecting the emission of the patches visited during the path.

Limiting the analysis to at most $D+1$ bounces, the solution of the rendering equation can be obtained as a $2D$-dimensional integral:

$$\mathbf{L}(\omega) = \left(\frac{1}{4\pi}\right)^D\cdot\int_{\Omega}\ldots\int_{\Omega}\left[\mathbf{L}^e(\omega) + 4\pi\cdot\mathbf{F}(\omega_1',\omega)\cdot\mathbf{I}_D\right]\;d\omega_D'\ldots d\omega_1'. \tag{11}$$

This high-dimensional integral can be estimated by Monte-Carlo or quasi-Monte Carlo quadrature. Note that here the integrand is continuous and piecewise continuously differentiable, which makes it of finite variation in the sense of Hardy and Krause. Thus we can really expect that the error of the quasi-Monte Carlo quadrature will be in the order of $O(\log^{2D-1} M/M)$ where M is the number of sample points [33].

2.4 Monte-Carlo or Quasi-Monte Carlo integration

In order to evaluate formula (11), M random or quasi-random [12] walks should be generated (the difference is that in Monte-Carlo walks the directions are sampled randomly, while in quasi-random walks they are sampled from a $2D$-dimensional low-discrepancy sequence, such as the $2D$-dimensional *Halton* or *Hammersley* sequences [19]). When the D-bounce irradiance is available, it is multiplied by the BRDF defined by the last direction ω_1 and the viewing direction ω to find a Monte-Carlo estimate of the radiance that is visible from the eye position. Note that this step makes the algorithm view-dependent.

There are basically two different methods to calculate the image estimate. On the one hand, evaluating the BRDF once for each patch, a radiance value is assigned to them, then in order to avoid "blocky" appearance, bi-linear smoothing can be applied. Using Phong interpolation, on the other hand, the radiance is evaluated at each point visible through a given pixel using the irradiance field, the surface normal and the BRDF of the found point. In order to speed up this procedure, the surface visible at each pixel, the visibility direction and the surface normal can be determined in a preprocessing phase and stored in a map. Phong interpolation is more time consuming but the generated image is not only numerically precise, but is also visually pleasing.

The final image is the average of these estimates. The complete algorithm — which requires just one variable for each patch i, the max d-bounce irradiance $\mathbf{I}[i]$ — is summarized in the following:

for $m = 1$ **to** M **do** *// samples of global walks*
  Generate $(\omega_1^{(m)}, \omega_2^{(m)}, \ldots, \omega_D^{(m)})$
  $\mathbf{I} = 0$
  for $d = 0$ **to** $D - 1$ **do** *// a walk*
    $\mathbf{I} = \mathbf{A}(\omega'_{D-d}) \cdot (\mathbf{L}^e(\omega'_{D-d}) + 4\pi \cdot \mathbf{F}(\omega'_{D-d+1}, \omega'_{D-d}) \cdot \mathbf{I})$
  endfor
  Calculate the image estimate from the irradiance $\mathbf{I}$
  Divide the estimate by M and add to the Image
endfor
Display Image

2.5 Improved walking techniques

The algorithm that has been derived directly from the quadrature formulae uses direction ω_1 to evaluate the contribution of the 1-bounces, directions (ω_1, ω_2) for the 2-bounces, $(\omega_1, \omega_2, \omega_3)$ for the 3-bounces, etc. This is just a

little fraction of the information that can be gathered during the complete walk. We could also use the samples of $\omega_1, \omega_2, \omega_3$, etc. to calculate the 1-bounce contribution, (ω_1, ω_2), (ω_1, ω_3), ..., (ω_2, ω_3), etc. combinations of directions for 2-bounces, etc. This is obviously possible, since if the samples of $(\omega_1, \omega_2, \ldots \omega_D)$ are taken from a uniform sequence, then all combinations of its elements also form uniform sequences in lower dimensional spaces.

last direction	1-bounce	2-bounce	3-bounce
3	$\mathbf{J}_0(\omega_3)$		
2	$\mathbf{J}_0(\omega_2)$	$\mathbf{J}_1(\omega_3, \omega_2)$	
1	$\mathbf{J}_0(\omega_1)$	$\mathbf{J}_1(\omega_3, \omega_1) + \mathbf{J}_1(\omega_2, \omega_1)$	$\mathbf{J}_2(\omega_3, \omega_2, \omega_1)$

Table 1
Irradiance matrix of a patch for $D = 3$

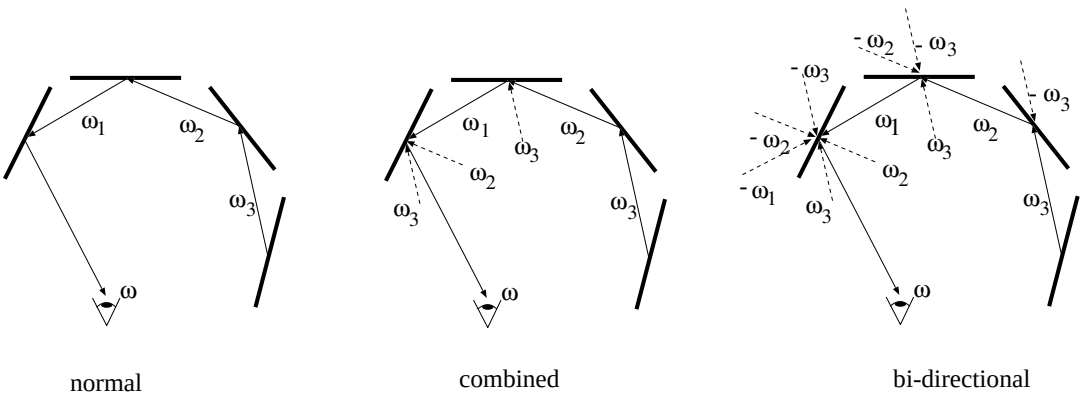

Fig. 3. Normal, combined and bi-directional walking technique

If all possible combinations are used, then each random walk generates $\binom{D}{d}$ samples for the d-bounces, which can be used to increase the accuracy of the method. Note that the increased accuracy of this "*combined*" method is for free in terms of additional visibility computation. However, due to the dependence of the BRDF functions on two directions and due to the fact that different bounces will be estimated by different numbers of samples, the required storage per patch is increased to $D(D+1)/2$ variables. Since D is 5 to 8 in practical cases, this storage overhead is affordable.

Now each patch is represented by a triangle matrix $\mathcal{I}$, where the (i, j) element stores the sum of those i-bounce irradiances where the last direction is ω_j. Table 1 shows an example for $D = 3$.

The complete combined algorithm is shown below:

for $m = 1$ **to** M **do**
    Generate $(\omega_1^{(m)}, \omega_2^{(m)}, \ldots, \omega_D^{(m)})$

for $d = 0$ **to** $D - 1$ **do** *// random or quasi-random walk*
$\mathcal{I}[1][D-d] = \mathbf{A}(\omega'_{D-d}) \cdot \mathbf{L}^e(\omega'_{D-d})$
for $b = 2$ **to** $d + 1$ **do**
$\mathcal{I}[b][D-d] = 0$
for $pd = b - 1$ **to** $d - 1$ **do**
$\mathcal{I}[b][D-d]$ += $4\pi \cdot \mathbf{A}(\omega^{(m)}_{D-d}) \cdot \mathbf{F}(\omega^{(m)}_{D-pd}, \omega^{(m)}_{D-d}) \cdot \mathcal{I}[b-1][D-pd]$
endfor
endfor
endfor
Divide the b-bounce estimates ($\mathcal{I}[b][d]$) by $\binom{D}{b}$
Calculate the image estimate from the $\mathcal{I}[b][d]$ estimates
Divide the estimate by M and add to the Image
endfor
Display Image

Furthermore, when the radiance is transferred to a direction, the required information to transfer the radiance to the opposite direction is also available, that is when computing the geometry matrix $\mathbf{A}(\omega')$ for some direction, the matrix for the reverse direction $\mathbf{A}(-\omega')$ is usually also known paying very little or no additional effort.

The improvement that takes advantage of this is called the *bi-directional* algorithm.

Due to the $\cos^*$ function only one of the elements $\mathbf{A}(\omega')_{ij}$ and $\mathbf{A}(-\omega')_{ij}$ can be non zero. It means that bi-directional techniques do not even require additional storage and a single geometry matrix can be used to store the values for both ω' and $-\omega'$. It can be decided whether a matrix element is valid for ω' or $-\omega'$ by inspecting the angle between its normal vector and the given directions.

Formally, in the bi-directional technique ω_1, $-\omega_1$, ω_2, $-\omega_2$, etc. are used to calculate the 1-bounce contribution, (ω_1, ω_2), $(\omega_1, -\omega_2)$, $(-\omega_1, \omega_2)$, $(-\omega_1, -\omega_2)$, (ω_1, ω_3), $(\omega_1, -\omega_3)$, etc. combinations of directions for 2-bounces, etc. This multiplies the number of samples used for the computation of 1-bounces by 2, for the 2-bounces by 4 and generally for the d bounces by 2^d, which can be quite significant.

The additional samples of the "combined" and particularly the "bi-directional" walking techniques increase the accuracy as shown by figure 4. The test scene was the homogeneous Cornell-box where all surfaces have constant 0.5 diffuse reflectance and emission, which allowed to solve the rendering equation analytically [12] (the solution is $L = 1 - 2^{-(D+1)}$) to find a reference for the error analysis. Note that although quasi-Monte Carlo sampling is generally better, the improvement provided by the combined and bi-directional methods is less

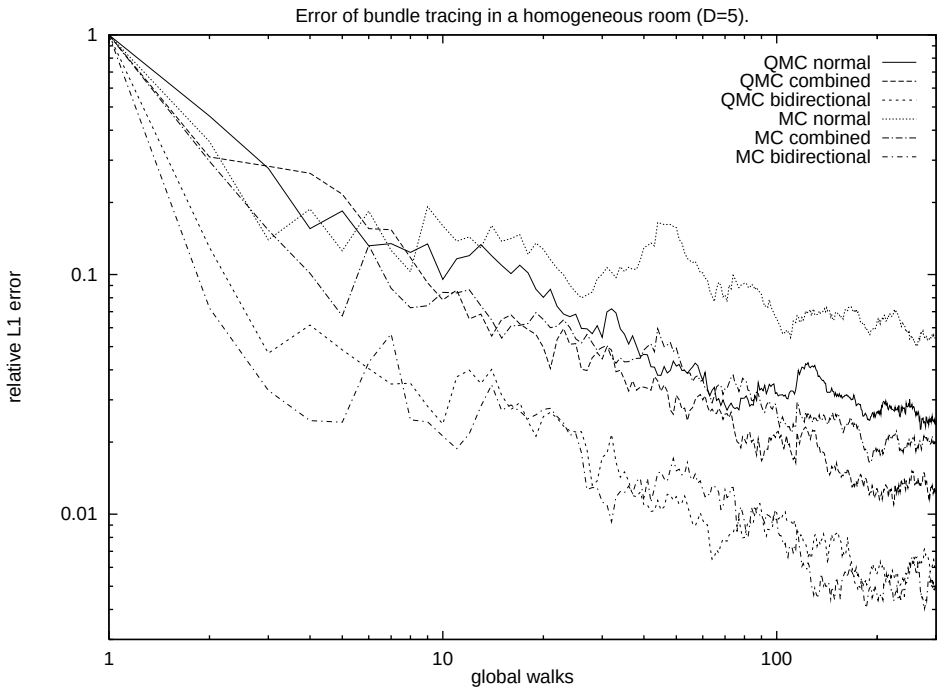

Fig. 4. Combined and bi-directional walking techniques versus normal walk

for the quasi-Monte Carlo walk than for the Monte-Carlo walk. This can be explained by the fact that the low-discrepancy points are so "well-designed" that mixing different sets of them does not improve the quadrature much further.

2.6 Making the global walk estimates unbiased

The methods introduced so far fall into the category of random walk techniques which calculate only the first D terms of the infinite Neumann series and simply ignore the rest. Consequently, the estimate will be biased.

However, the bias can be easily eliminated using a simple correction of the emission function $\mathbf{L}^e$ when calculating higher order interreflections. Note that a global walk provides random estimates for the following terms:

$$\mathbf{L}^e + \mathcal{T}\mathbf{L}^e + \ldots + \mathcal{T}^D\mathbf{L}^e.$$

Thus having computed the first walk, we also have an estimate for $\mathcal{T}^D\mathbf{L}^e$ represented by D-bounce irradiance $\mathbf{J}_D$. Let us use this estimate to correct the emission function in the higher order terms when the second global walk is computed:

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

$$\mathbf{L}^e + \mathcal{T}\mathbf{L}^e + \ldots + \mathcal{T}^D\mathbf{L}^e + \mathcal{T}^{D+1}\mathbf{L}^e + \ldots + \mathcal{T}^{2D}\mathbf{L}^e. \qquad (12)$$

This gives us estimates not only for the bounces from 0 to D but also for the bounces from $D+1$ to $2D$. The D bounce irradiance will store an estimate for $\mathcal{T}^D\mathbf{L}^e + \mathcal{T}^{2D}\mathbf{L}^e$, which can again be used to compensate the emission. Thus after the second step we have estimates for the 0 to $3D$ bounces. Asymptotically, this method will generate estimates for all bounces. However, if M global walks are computed, then the number of estimates for bounces of 0 to D is M, for bounces of $D+1$ to $2D$ is $M-1$, for bounces $2D+1$ to $3D$ is $M-2$ etc., which still results in some small energy defect.

Note that care should be taken when this method is applied with low-discrepancy sequences since here asymptotically infinite-dimensional integrals are approximated using $2D$-dimensional low-discrepancy sequences. The 1-uniform [13] Halton and the Hammersley sequences are not appropriate, thus they can introduce artifacts, especially if D is low. Note that this problem is specific to quasi-Monte Carlo integration and does not occur when the classical Monte-Carlo method is used to select the sample points (a random sequence is ∞-uniform [13]).

3 Calculation of visible areas

A crucial part of the method is the determination of areas $A(i, j, \omega')$ that express the projected area of patch j that is visible from patch i in direction ω'.

For a given i, it requires the solution of a visibility problem assuming the eye position in patch i and the viewing direction to be ω'. This fixed direction is called the transillumination direction [16]. The visibility problem identifies those regions of patch i from where patch j is visible. The projected areas of regions of different js should then be computed and summed to generate the needed function $A(i, j, \omega')$. Since $A(i, j, \omega')$ is needed for all i and j values in a fixed direction, this is a global visibility problem where the viewing direction is fixed but the eye position visits all patches. It would not be wise if we solved this problem handling different patches independently, but the patches should be processed parallely and the knowledge gained when the viewing position is at one patch must be reused for other patches as well. Theoretically, we could use either discrete or continuous visibility algorithms. Discrete algorithms that solve the visibility problem for a finite number of pixels are faster and the built-in rendering hardware and z-buffer can also be exploited, but in order to handle all patches simultaneously, the "window" of the algorithm should include all patches. The large window, however, should be decomposed into sufficiently small pixels to provide the required precision, which might result in an unacceptable number of pixels. Continuous algorithms are free from these resolution problems, but are usually more difficult to implement and are

slower.

Taking into account that the viewing direction determines which patch can be in front of another one, the first step of such an continuous algorithm is the sorting of patches in the given viewing direction. Such a sorting is not obvious if the patches overlap in this direction, but this can be solved by the clipping proposed in the painter's algorithms [18].

In order to calculate $A(i, j, \omega')$ for patch i, only those j patches should be considered that are in front of patch i in the generated order. From the projection of these patches onto a plane perpendicular to the transillumination direction, a planar graph is constructed where each region identifies a patch that is visible in it or specifies that no patch is visible here. Such a planar graph is usually called the *visibility map*.

The algorithm processes the patches in the order defined by the transillumination direction and maintains the visibility graph dynamically [33].

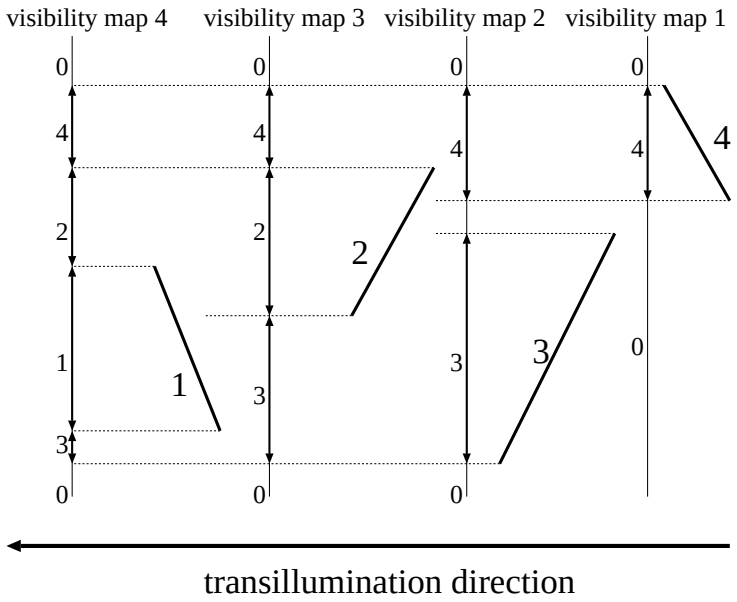

Fig. 5. Local visibility maps

When the processing of patch i is started, the visibility map shows which patches are visible from patch i. To calculate the $A(i, j, \omega')$ values, those patches which have projections either entirely or partly in the projection of patch i are selected from the visibility map, and are clipped onto patch i to clearly separate inner regions. The process of clipping of the patches onto each other is quite similar to the Weiler-Atherton visibility algorithm [39]. Then the projected areas of those patch parts which are inside the projection of patch i are summed to find the $A(i, j, \omega')$ values.

When we step onto the patch next to patch i, a new visibility map is created

by replacing those region parts that are inside the projection of A_i by the projection of A_i. Then, if patch i can reflect energy onto the next patch (it is a back facing patch with respect to the transillumination direction), then patch i should be added to the visibility map, otherwise, the place of the projection of patch i will be empty.

The algorithm, that maintains a list P for the sorted patches, a list V for the projected patches that are currently in the visibility map, a list O for those clipped, not-hidden patches whose projections are outside patch i, and a list I for those clipped, not-hidden patches whose projections are inside patch i, and computes the area of projected territory j by function $A(j)$ is as follows:

list P = Sort patches in direction ω'
visibility map $V = \{\ \}$
for each patch i in P **do**
    Clip patches in V onto patch i and
    generate: O = outside list, I = inside list
    if patch i is front facing **then**
        $A(i, j, \omega') = \sum_{j \in I} A(j)$
        $V = O$
    else
        $V = O$ + patch i
    endif
endfor

If the number of patches is n, then the size of the visibility map is $O(n)$ in the average case but $O(n^2)$ in the worst-case. Thus the resulting algorithm that compares each patch with the actual visibility map will have $O(n^2)$ average case and $O(n^3)$ worst case time-complexity. This is not acceptable when complex scenes are processed. In order to reduce the complexity, we can use a wide selection of computational geometry methods that usually apply spatial decomposition on the plane to reduce the number of unnecessary comparisons [34].

Another alternative is to introduce randomization (or quasi-randomization) in the computation of the visible areas, as suggested by Russian roulette. When the outside list O is generated, the patches that have small products of area times radiance are randomly (or quasi-randomly) deleted from the list. Suppose that those patches are considered for deletion for which $A(j) \cdot L_j < \epsilon$ where $A(j)$ is the area of the region j and L_j is the radiance of the associated patch, and let us define the deletion probability to be proportional to $A(j) \cdot L_j$. Whenever a small patch is kept, its radiance is divided by $A(j) \cdot L_j/\epsilon$ to compensate for occasions when it is deleted.

Thus the following instructions should be inserted in the algorithm:

for each region j in O **do**
  if $A(j) \cdot L_j < \epsilon$ **then**
    Generate a random number r in $[0 \ldots 1]$
    if $A(j) \cdot L_j < r \cdot \epsilon$ **then** Delete patch i from O
    else $L_j = \epsilon / A(j)$
  endif
endfor

In the context of Monte-Carlo methods, this corresponds to the Russian roulette. In quasi-Monte Carlo integration r is an additional integration variable, that should be generated by an independent low-discrepancy sequence.

4 Simulation results

The presented algorithm has been implemented and tested. During simulations, the directions were generated from both quasi-Monte Carlo and from Monte-Carlo sequences.

Fig. 6. A scene of a Beethoven and a teapot rendered by 500 Monte-Carlo walks

Figure 6 shows a scene as rendered after 500 walks and Figure 7 after 3000 walks when the algorithm is fully converged. We used the extension that made the global walks unbiased. The scene contains specular, metallic objects tessellated to 9519 patches. The calculation of the left image took 9 minutes on

Fig. 7. A scene of a Beethoven and a teapot rendered when fully converged

a SGI O2 computer. This pair of images demonstrates that this algorithm can provide good image quality even after a relatively low number of walks.

5 Conclusions

This paper presented a global light-tracing method which generates quasi-random walks and follows all light interactions that are parallel with the directions of the steps of the walk.

This method can solve the non-diffuse radiosity problem — that is the general rendering equation — without increasing the number of free variables and the required storage as happens in earlier methods. Since in the finite-element phase the patches are assumed to have position independent radiance distribution, and in the random-walk phase the directional integrals are approximated by quasi-Monte Carlo quadrature, the method is accurate if the radiance function is quite smooth (perfect mirrors, for example, can produce artifacts).

6 Acknowledgments

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] L. Aupperle and P. Hanrahan. A hierarchical illumination algorithms for surfaces with glossy reflection. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 155–162, 1993.

[2] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical Monte-Carlo radiosity. In *Rendering Techniques '98*, pages 259–268, 1998.

[3] C. Buckalew and D. Fussell. Illumination networks: Fast realistic rendering with general reflectance functions. *Computer Graphics (SIGGRAPH '89 Proceedings)*, 23(3):89–98, July 1989.

[4] P. H. Christensen, D. Lischinski, E. J. Stollnitz, and D. H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, 16(1):3–33, 1997.

[5] P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics*, 15(1):37–71, 1996.

[6] Michael Cohen and Donald Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, pages 31–40, 1985.

[7] R. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, pages 137–145, 1984.

[8] 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.

[9] P. Hanrahan, D. Salzman, and L. Aupperle. Rapid hierachical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 1991.

[10] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 133–142, 1986.

[11] J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 143–150, 1986.

[12] 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.

[13] D.E. Knuth. *The art of computer programming. Volume 2 (Seminumerical algorithms)*. Addison-Wesley, Reading, Mass. USA, 1981.

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

[15] 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.

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

[17] L. Neumann, A. Neumann, and P. Bekaert. Radiosity with well distributed ray sets. *Computer Graphics Forum (Eurographics'97)*, 16(3):261–270, 1997.

[18] M. E. Newell, R. G. Newell, and T. L. Sancha. A new approach to the shaded picture problem. In *Proceedings of the ACM National Conference*, pages 443–450, 1972.

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

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

[21] M. Sbert. *The Use of Global Directions to Compute Radiosity.* PhD thesis, Catalan Technical University, Barcelona, 1996.

[22] M. Sbert. Error and complexity of random walk Monte-Carlo radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 3(1), 1997.

[23] M. Sbert, R. Martinez, and X. Pueyo. Gathering multi-path: a new Monte-Carlo algorithm for radiosity. In *Winter School of Computer Graphics '98*, pages 331–338, Plzen, Czech Republic, 1998.

[24] M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer. Global multipath Monte Carlo algorithms for radiosity. *Visual Computer*, pages 47–61, 1996.

[25] P. Schröder, S.J. Gortler, M.F. Cohen, and P. Hanrahan. Wavelet projections for radiosity. *Computer Graphics Forum*, 13(2):141–151, 1994.

[26] P. Shirley. A ray-tracing method for illumination calculation in diffuse-specular scenes. In *Proc. Graphics Interface*, pages 205–212, 1990.

[27] P. Shirley. Time complexity of Monte-Carlo radiosity. In *Eurographics '91*, pages 459–466. Elsevier Science Publishers, 1991.

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

[29] F. Sillion and Puech C. *Radiosity and Global Illumination.* Morgan Kaufmann Publishers, Inc., San Francisco, 1994.

[30] F. Sillion, G. Drettakis, and C. Soler. Clustering algorithm for radiance calculation in general environments. In *Rendering Techniques '95*, pages 197–205, 1995.

[31] I. Sobol. *Die Monte-Carlo Methode.* Deutscher Verlag der Wissenschaften, 1991.

[32] M. Stamminger, Slussalek P., and H-P. Seidel. Three point clustering for radiance computations. In *Rendering Techniques '98*, pages 211–222, 1998.

[33] 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.

[34] L. Szirmay-Kalos, T. Fóris, and W. Purgathofer. Quasi-Monte Carlo global ray-bundle tracing with infinite number of rays. In *Winter School of Computer Graphics '98*, pages 386–393, Plzen, Czech Republic, 1998.

[35] E. Veach and L. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Rendering Techniques '94*, pages 147–162, 1994.

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

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

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

[39] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. In *Computer Graphics (SIGGRAPH '77 Proceedings)*, pages 214–222, 1977.

[40] K. Zimmerman and P. Shirley. A two-pass solution to the rendering equation with a source visibility preprocess. In *Rendering Techniques '95*, pages 284–295, 1995.