EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (*2001*), *Number 3*

Global Illumination as a Combination of Continuous Random Walk and Finite-Element Based Iteration

László Szirmay-Kalos, Ferenc Csonka, György Antal

Department of Control Engineering and Information Technology,
Technical University of Budapest,
Budapest, Pázmány Péter s. 1/D, H-1117, HUNGARY
Email: szirmay@iit.bme.hu

Abstract

The paper introduces a global illumination method that combines continuous and finite-element approaches, preserving the speed of finite-element based iteration and the accuracy of continuous random walks. The basic idea is to decompose the radiance function to a finite-element component that is only a rough estimate and to a difference component that is obtained by Monte-Carlo techniques. Iteration and random walk are handled uniformly in the framework of stochastic iteration. This uniform treatment allows the finite-element component to be built up adaptively aiming at minimizing the Monte-Carlo component. The method is also suited for interactive walkthrough animation in glossy scenes since when the viewpoint changes, only the small Monte-Carlo component needs to be recomputed.

1. Introduction

Global illumination algorithms, which aim at the physically correct simulation of the light propagation, solve the rendering equation

$$L = L^e + \mathcal{T}_{f_r} L,$$

which expresses the radiance $L(\vec{x}, \omega)$ of point $\vec{x}$ at direction ω as a sum of the emission L^e and the reflection of all point radiances that are visible from here. The reflection of the radiance of visible points is expressed by an integral operator

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

which is also called as the *light transport operator*. In this equation h is the visibility function finding that point which is visible from $\vec{x}$ at direction $-\omega'$, f_r is the BRDF and θ' is the angle between the surface normal and direction $-\omega'$.

The solution of the rendering equation and the computation of an image from the radiance of the points visible in different pixels are rather time consuming. The timing requirements become even more prohibitive when animation sequences are needed. The computation time can be reduced if the similarity or coherence of the radiance function in a single frame and even in multiple frames in the sequence are exploited. It means that the radiance of neighboring points in an image or in subsequent frames in the animation are quite close thus a great portion of the illumination and visibility information can be reused during the solution.

Global illumination algorithms can be classified as random walk and iteration techniques.

Random walk algorithms search light paths following a *depth-first* strategy[15, 9, 5, 12, 27]. From mathematical point of view, they are based on the Neumann series expansion of the rendering equation and compute the color of a pixel as

$$C = \sum_{i=0}^{\infty} \mathcal{M} \mathcal{T}_{f_r}^i L^e, \tag{1}$$

where $\mathcal{M}$ is the measurement operator finding the average radiance of the points visible in this pixel. The terms of this series are ever increasing high-dimensional integrals that are estimated by Monte-Carlo quadrature in order to avoid the exponential core of classical quadrature rules[21]. Monte-Carlo quadrature takes randomly or quasi-randomly[10] selected discrete samples in the domain of the possible light-paths, evaluate their contribution to the camera and obtain the final image as the average of these contributions. The

convergence of Monte-Carlo quadrature is in the order of $\mathcal{O}(m^{-0.5})$, where m is the number of light paths. Quasi-Monte Carlo quadrature is faster, but for infinite dimensional domains and for the infinite variation integrand of the rendering equation, the order of convergence is worse than $\mathcal{O}(m^{-(1-\varepsilon)})$ that could be predicted for smooth integrands by the Koksma-Hlawka inequality[23]. Since just discrete samples of the radiance and the geometry are needed, these methods can work with the original geometry and require no tessellation. The samples are generated independently, thus this approach is free from error accumulation and can be easily ported to parallel machines. The obtained result will be the asymptotically correct solution of the original problem. Unfortunately, we have to pay a high-price for this asymptotically correct solution. Since the paths are generated independently, the earlier results cannot be efficiently stored and reused in the computations. On the one hand, these methods are unable to efficiently utilize the space and time coherence of the radiance function. An exception is the Metropolis light transport[26] which obtains the new path by perturbing the last path rather than rebuilding it from scratch. Although remembering just the last path is a very limited knowledge of the previous paths, this trick can result in significant performance improvements in scenes containing highly specular materials or allowing just a small fraction of paths to have non-zero contribution. On the other hand, queries to the geometry are unstructured, thus instead of efficient visibility methods random walk algorithms must use ray-shooting. Note also that in equation (1) the measurement operator that depends on the camera is included in all terms, thus this approach is strongly view dependent. If the camera changes, the complete calculation should be started from scratch. In their original form, random walk methods are unable to utilize any coherence among frames thus they cannot be used in fast animation sequences.

Iteration techniques, on the other hand, generate light paths according to a *breadth-first* search[4, 3]. In a single step all paths are advanced once simultaneously. These techniques are based on the fact that the solution of the rendering equation is the fixed point of the following iteration scheme:

$$L(m) = L^e + \mathcal{T}_{fr} L(m-1).$$

If this scheme is convergent, then the pixel colors can be obtained as a limiting value:

$$C = \lim_{m \to \infty} \mathcal{M} L(m).$$

Iteration converges with the speed of a geometric series, i.e. the error from the limiting value is in the order of $\mathcal{O}(a^m)$ where a is the contraction of integral operator $\mathcal{T}_{fr}$. The contraction is proportional to the average albedo of the surfaces and depends on how open the scene is. Note that iteration uses the estimate of the complete radiance function, thus it can potentially exploit coherence and reuse previous information, and can optimize geometric queries allowing fast and hardware supported visibility algorithms. Since the complete radiance function is inserted into the iteration formula, parallelization is not as trivial as for random walks, and the error introduced in each step may accumulate to a large value[23]. To store the radiance estimates, finite-element approaches should be used which represent the radiance function in a finite function series form:

$$L(\vec{x}, \omega) = \sum L_j \cdot b_j(\vec{x}, \omega)$$

where functions $b_j(\vec{x}, \omega)$ are pre-defined basis functions and parameters L_j are scalars. Basis functions $b_j(\vec{x}, \omega)$ are usually decomposed to a product of positional ($s_k(\vec{x})$) and directional basis functions ($d_i(\omega)$). The positional basis functions may be either constant or linear on a patch, while the directional basis functions can also be piece-wise constant[6], spherical harmonics[16] or Haar functions[19]. Due to the fact that the radiance has 4 variates and changes quickly, an accurate finite-element representation requires very many basis functions, which makes these algorithms both storage and time consuming. If the number of basis functions is less than necessary, light-leaks may occur and the shadows and highlights may be placed incorrectly[17]. Unlike in random walks, the radiance estimates $L(m)$ are completely view-independent, thus when they are available, the image can be obtained from any viewpoint. Thus iteration can potentially exploit the coherence of frames. However, it has a high prize in terms of storage space.

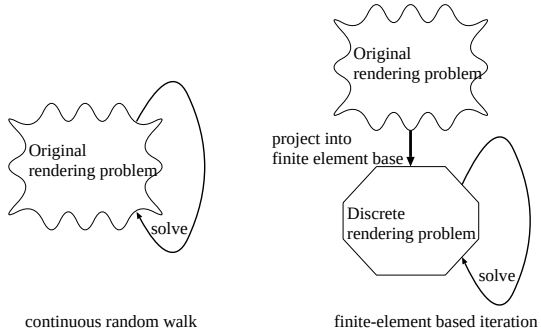


Figure 1: *Comparison of the conceptual schemes of the continuous random walk and iteration techniques*

Comparing random walk and iteration we can conclude that random walk requires just one light-path to be stored while iteration needs very many variables, but random walk uses practically no coherence information while iteration can strongly exploit it. Iteration is slow due to the handling of the very many finite elements, while random walks are slow due to the lack of the utilization of the coherence. Although a single iteration step requires much more computation than a single random light-path, the $\mathcal{O}(a^m)$ convergence of iteration still seems to be far superior to the $\mathcal{O}(m^{-0.5})$ convergence of random walks. However, random walk converges to the real solution while iteration to the solution of the finite-element approximation of the original problem (figure 1).

Furthermore, if the light-transport operator is not exactly evaluated, the limiting value is also distorted by the cumulative error. Thus only the initial behavior of iteration overcomes random walk. Finite-element methods are good when the radiance is smooth, i.e. for diffuse or glossy scenes. Random walk, on the other hand, is effective when BRDF sampling can significantly reduce the candidates of directions, that is when the surfaces are highly specular. Since the two approaches can complement each other, their combination is a promising alternative. Multi-pass[28, 20] approaches separately run different algorithms being good in finding different types of light-paths, and combine their results. Two-pass methods, on the other hand, store the result of the first-phase in some approximation form, which is then used by the view-dependent random walk second phase. The information of the first phase can be a radiosity solution[28], a photon-map[8], irradiance vectors[1], i.e. incomplete light-paths that are completed by the second phase. Alternatively, the result of the first phase can be some importance information that is used later to guide the walks towards important regions[7, 24]. A common problem of these methods is that the computational times given to the preprocessing phase and to the final gather should be decided before starting the rendering and the different approaches cannot strengthen each other on the fly. Another interesting combination of the random walk and iteration methods occurs in multi-path[14] algorithms. Since they were designed to attack the diffuse radiosity problem, the original problem is projected to a finite-element base, where it is solved by a special random walk, which simultaneously advances several but not all light-paths.

2. The combined approach

In this paper we present a combined approach which tries to get the benefits from both approaches in a single pass. Intuitively, iteration is used only for those components that can be stored in a simple way but which are responsible for the greater part of the radiance function. Components that would have expensive finite-element representation, on the other hand, are estimated by Monte-Carlo method on the fly. The combined method can be thought of as an adaptation strategy that automatically subdivides the original global illumination problem into a simple finite-element problem and a low-variance Monte-Carlo estimation problem. The Monte-Carlo part is responsible for building up the finite-element part in order to keep itself relatively small. The finite-element part, in turn, reduces the variance of the Monte-Carlo integration (figure 2). In this way, the finite-element method and the Monte-Carlo simulation help each other. The accuracy and the resolution of the finite-element representation is not important, since it is only used as a rough estimate that is corrected by the Monte-Carlo simulation. Due to this and to the adaptive evolution of the finite-element decomposition, accurate results can be obtained with relatively few basis functions. The separation of Monte-Carlo and the finite-element parts seems to be similar to a classical variance reduction technique called the separation of the main part[18]. However, here the main part is not known in advance, neither can it analytically be integrated. In [11] this problem has been solved by function approximation. Here the main part is generated adaptively and is integrated by a Monte-Carlo quadrature rule simultaneously with the calculation of the Monte-Carlo component.

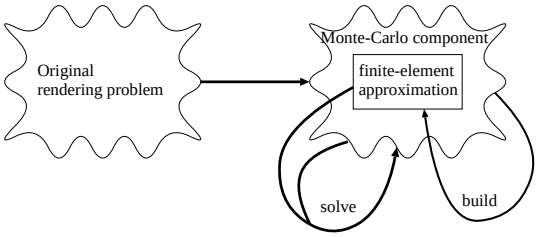


Figure 2: *Conceptual scheme of the combined algorithm*

In order to work out the details, a formal framework is needed that can incorporate both continuous random walks and finite-element based iterations. The formal basis is the stochastic iteration, which was originally proposed for the solution of the linear equations obtained in the radiosity setting[13, 14, 2], then extended for the continuous rendering equation[22]. It means that in the iteration sequence a random transport operator is used instead of the light-transport operator, which gives back the light-transport operator in the average case:

$$L(m) = L^e + \mathcal{T}_{f_r}^* L(m-1), \qquad E[\mathcal{T}_{f_r}^* L] = \mathcal{T}_{f_r} L. \qquad (2)$$

The pixel colors are computed as an average of the estimates of all iteration steps

$$C(m) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathcal{M}L(i) = \frac{1}{m} \cdot \mathcal{M}L(m) + \left(1 - \frac{1}{m}\right) \cdot C(m-1).$$

Note that stochastic iteration can find a flexible compromise between finite-element based iteration and random walks. If the random light transport operator is deterministic, then we get back the classical iteration. However, if it is randomized to a degree that it uses the radiance function just in a single point and single direction, then it gives back the random walk[22].

2.1. Decomposition of the radiance function

Let us decompose the radiance function L to an emission L^e, to a reflected component $\tilde{L}$ that can be approximated by the linear combination of the finite-elements (called the *finite-element component*), and to a reflected residuum $\Delta L(\omega)$ (called the *Monte-Carlo component*) that is estimated by Monte-Carlo simulation:

$$L = L^e + \tilde{L} + \Delta L. \qquad (3)$$

Suppose that the positional and directional basis functions and the adjoint basis functions are $s_i(\vec{x}), d_j(\omega)$ and $\tilde{s}_i(\vec{x}), \tilde{d}_j(\omega)$, respectively, thus the component approximated by the finite-elements is

$$\tilde{L}(\vec{x},\omega) = \sum_i \sum_j s_i(\vec{x}) d_j(\omega) \cdot \mathbf{L}_{ij}, \qquad \mathbf{L}_{ij} = \langle L - L^e, \tilde{s}_i \tilde{d}_j \rangle,$$

where

$$\langle u, v \rangle = \int_S \int_\Omega u(\vec{x},\omega) \cdot v(\vec{x},\omega) \cdot \cos\theta \; d\omega d\vec{x}$$

is the scalar product of two functions.

Let us substitute this decomposition into the stochastic iteration formula (equation 2):

$$L(m) = L^e + \mathcal{T}^*_{f_r}(L^e + \tilde{L}(m-1) + \Delta L(m-1)).$$

The finite-element component is the projection into the adjoint base, that is

$$\tilde{L}(m) = \sum_i \sum_j s_i(\vec{x}) d_j(\omega) \cdot \langle L(m) - L^e, \tilde{s}_i \tilde{d}_j \rangle.$$

Since f_r is the only term that depends on the output direction ω, we can further obtain:

$$\tilde{L}(m) = \sum_i s_i(\vec{x}) \cdot \int_S \mathcal{T}^*_{\tilde{f}_r} L(m-1) \cdot \tilde{s}_i(\vec{x}) \; d\vec{x}$$

where $\tilde{f}_r$ is the projected BRDF:

$$\tilde{f}_r(\omega', \vec{x}, \omega) = \sum_j d_j(\omega) \cdot \int_\Omega f_r(\omega', \vec{x}, \omega) \cdot \tilde{d}_j(\omega) \cdot \cos\theta \; d\omega.$$

Suppose that piece-wise constant basis functions are used, i.e. $s_i(\vec{x})$ and $\tilde{s}_i(\vec{x})$ are 1 and $1/A_i$ respectively if $\vec{x}$ is in patch i and zero otherwise, and $d_j(\omega)$ and $\tilde{d}_j(\omega)$ are 1 and $1/\Omega^*_j$ respectively if ω is in solid angle Ω_j and zero otherwise, where $\Omega^*_j = \int_{\Omega_j} \cos\theta \; d\omega$. These basis functions result in the following formulae:

$$\tilde{L}(m) = \sum_i s_i(\vec{x}) \cdot \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}^*_{\tilde{f}_r} L(m-1) \; d\vec{x} \qquad (4)$$

where

$$\tilde{f}_r(\omega', \vec{x}, \omega) = \sum_j d_j(\omega) \cdot \frac{1}{\Omega^*_j} \cdot \int_{\Omega_j} f_r(\omega', \vec{x}, \omega) \cdot \cos\theta \; d\omega.$$

The projected BRDF $\tilde{f}_r$ can be computed in the preprocessing phase for each possible material and stored in tables or can be estimated on the fly. The values should not be accurate, since the result is corrected by the algorithm. Note that if a single directional basis function is used, then the projected BRDF becomes the *albedo* divided by π.

The Monte-Carlo component can be obtained by subtracting the emission and the finite-element component from the reflected radiance:

$$\Delta L(m) = L - L^e - \tilde{L} = \mathcal{T}^*_{f_r} L(m-1) - \sum_i s_i(\vec{x}) \cdot \int_S \mathcal{T}^*_{\tilde{f}_r} L(m-1) \cdot \tilde{s}_i(\vec{x}) \; d\vec{x}.$$

This Monte-Carlo component should be stored until it is substituted into the iteration formula in the next step.

2.2. The new algorithm

The new algorithm follows a stochastic iteration scheme and in each iteration step the radiance is projected to the adjoint base and an image estimate is computed from the actual radiance. Note that the image estimates and the projected radiance values, as stochastic iteration in general, will not converge, but they will fluctuate around the real solution. Thus the final image is obtained as the average of these image estimates, and the finite-element component as the average of the projected radiance approximations. If the finite-element projection of the radiance at step m is $\tilde{L}'(m)$, then the finite-element part may be derived as follows:

$$\tilde{L}(m) = \frac{1}{m} \cdot \sum_{n=1}^{m} \tilde{L}'(n) = \frac{1}{m} \cdot \tilde{L}'(m) + \left(1 - \frac{1}{m}\right) \cdot \tilde{L}(m-1). \qquad (5)$$

The Monte-Carlo component, which is obtained as a difference between the actual radiance estimate and its finite-element projection, is used to correct the finite-element approximation. This also means that the behavior of the Monte-Carlo component in the support of a given basis function shows how accurately this basis function represents the radiance function and whether or not it should be refined. To exploit this fact, the iteration is broken down to phases and in each phase the norm of the Monte-Carlo components for all basis function domains are evaluated. Then using these norms, the finite-element structure is refined and patches and solid angles are broken down appropriately.

The complete algorithm is:

CombinedGlobalIllumination
  $\tilde{L}(0) = 0, \Delta L(0) = 0$
  for $m = 1$ **to** M **do**
    $L^r = \mathcal{T}^*_{f_r}(L^e + \tilde{L}(m-1) + \Delta L(m-1))$
    $\tilde{L}'(m) =$ projection of L^r to an adjoint base
    $\Delta L(m) = L^r - \tilde{L}'(m)$
    $\tilde{L}(m) = 1/m \cdot \tilde{L}'(m) + (1 - 1/m) \cdot \tilde{L}(m-1)$
    $C'(m) = \mathcal{M}(L^e + \tilde{L}(m) + \Delta L(m))$
    $C(m) = 1/m \cdot C'(m) + (1 - 1/m) \cdot C(m-1)$
    Contribute to the norms
    if m is an end of the phase
      Refine the finite-element structure
    endif
  endfor
  Display $C(m)$ colors
end

The dataflow of the new algorithm is shown in figure 3.

Note that the new reflected radiance $\tilde{L}(m)+\Delta L(m)$ is computed from the radiance generated by the random transport operator as first subtracting its finite-element projection then adding the average of these finite-element projections. At the beginning of the execution of the algorithm this replaces a high-variance main part by its estimated average, which is responsible for good initial convergence. Later, when the algorithm converges, the expected finite-element component gets close to its average, thus subtraction and addition compensate each other and the finite-element approximation does not distort final result. We could get the speed of the iteration together with the asymptotic accuracy of random walks.

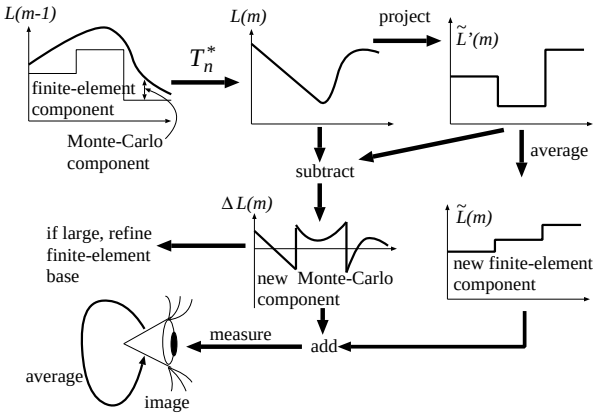


Figure 3: *Dataflow in the new algorithm*

This is a generic algorithm from which different specific versions can be built by inserting the random transport operator, the system of basis functions and the refinement oracle. Having selected the random transport operator and the family of basis functions, two sub-problems should be considered. We have to represent Monte-Carlo component $\Delta L(m)$ temporarily and compute its reflection to obtain the image estimate and the radiance in the next iteration cycle. On the other hand, refinement criteria are needed that are based on the norm of the Monte-Carlo component in the support of the basis functions.

3. Iterating with parallel radiance transfers

In this section a specific algorithm is discussed that transfers the radiance of all patches to a randomly selected global direction in each iteration cycle. The basis functions will be piece-wise constant and when refinement is necessary, the support of the given basis function is divided into four equal areas. However, unlike other deterministic iteration techniques, this finite-element representation does not aim at the accurate representation of the radiance, it is only for a rough approximation that is corrected by the radiance component obtained by Monte-Carlo simulation. On the other hand, the number of basis functions are not determined a-priori. The algorithm is started with a single constant basis function per patch which is refined on the fly if the Monte-Carlo component turns out to be too big on this patch.

Since the algorithm transfers the radiance into a randomly selected direction ω', the random transport operator is

$$L^r(\vec{x},\omega) = \mathcal{T}_{f_r}^* L = 4\pi \cdot L(h(\vec{x},-\omega'),\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \cos\theta'.$$

Indeed, if the direction is sampled uniformly, then its probability density is $1/4\pi$, thus the expectation of the random transport operator gives back the effect of the light transport operator $\mathcal{T}_{f_r} L$, as required by equation (2):

$$E[\mathcal{T}_{f_r}^* L] = \int_{\Omega'} 4\pi \cdot L(h(\vec{x},-\omega'),\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \cos\theta' \cdot \frac{d\omega'}{4\pi}.$$

The radiance transfer needs the identification of those points that are mutually visible in the global direction. In order to solve this global visibility problem, a window is placed perpendicular to the global direction. The window is decomposed into a number of pixels. A pixel is capable to store a list of patch indices and z-values. The lists are sorted according to the z-values. The collection of these pixels are called the *transillumination buffer*[13]. The patches are rendered one after the other into the buffer using a modified z-buffer algorithm which keeps all visible points not just the nearest one. Traversing the generated lists the pairs of mutually visible points can be obtained. For each pair of points, the radiance transfer is computed and the transferred radiance is multiplied by the BRDF, resulting in the reflected radiance L^r.

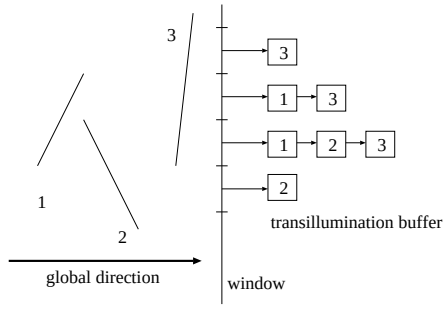


Figure 4: *Organization of the transillumination buffer*

From the reflected radiance the finite-element component can be obtained by a simple averaging operation according to equation (4). Note that if the integral is evaluated on the window, then the cosine factor is compensated:

$$\tilde{L}(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}_{\tilde{f}_r}^* L(m-1)\, d\vec{x} \approx$$

$$\frac{4\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P) \cdot \tilde{f}_r(\omega', P, \omega)$$

where P runs on the pixels covering the projection of patch i, $L^{in}(P)$ is the radiance of the surface point visible in pixel P, $f_r(\omega', P, \omega)$ is the BRDF of that point which receives this radiance coming through pixel P and δP is the area of the pixels.

It is straightforward to extend the method to be bi-directional, which transfers the radiance not only into direction ω' but also to $-\omega'$. Note that this does not even require additional visibility computation.

3.1. Temporary representation of the reflected component

The Monte-Carlo component $\Delta L(m)$ should temporarily be represented until it is included in the image estimate and is used to obtain the reflected radiance of the consecutive iteration step. In this section two alternatives are considered, the first one is an image-space approach while the second is an object-space technique.

3.1.1. Representing the radiance wavefront in the transillumination buffer

Since the radiance is transferred in a single direction through a discretized window, called the transillumination buffer, snapshots of this discretized window can also be used for the temporary representation of the radiance. Looking at figure 4, we can note that each pixel stores a list of patches. For each patch of the list, the visible patch is either the previous or the next member of this list. Identifying the visible patch, the radiance arriving through the given pixel at the given patch can be calculated. If we are interested in the reflected radiance of a point on a triangle, then this point is projected onto the window surface to select that pixel which stores the respective list needed for the incoming radiance information. From the incoming radiance, the outgoing radiance for any possible direction can be obtained by multiplying with the local BRDF.

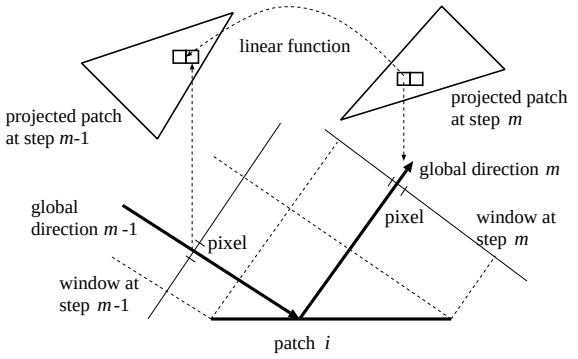


Figure 5: *Reflection of the radiance wavefront of a ray-bundle*

These calculations can be made fast using the incremental concept. Suppose that the outgoing radiance is required for the points of a triangle to compute the radiance transfer of the image estimates. The triangle is projected onto a pixel grid, which either corresponds to the image or to the transillumination buffer of the next iteration. Let us call this grid as the output grid to distinguish from the transillumination buffer which serves as the input buffer. The correspondence of the coordinates of the input and output buffers is linear if parallel projection is used and homogeneous linear for perspective projection. Note that the projection onto the transillumination buffer is always parallel, while the projection onto the image plane may be perspective. The operation is basically equivalent to texture mapping, where the input buffer serves as the texture.

3.1.2. Representing the radiance wavefront on sub-patches

The sources of the radiance are the patches, thus they can also be used to represent the temporary radiance in the global direction. Since the output radiance of the patches is not necessarily constant, patches are further subdivided until a sub-patch can be assumed to have constant radiance. Thus the radiance wavefront can also be stored as the radiance values of the collection of sub-patches.

Comparing the sub-patch method to the transillumination buffer representation we can note that the sub-patch method works well with lower resolution transillumination buffers but require fine geometric tessellation. It does not completely get rid of the finite-element approximation error, only reduces it. On the other hand, the transillumination buffer representation can process roughly tessellated surfaces, but needs high-resolution transillumination buffer. Lower resolution transillumination buffers might cause aliasing artifacts. The random nature of the algorithm eliminates these artifacts in the final image, but the phenomenon can slow down the convergence. The number of BRDF calculations is proportional to the number of transillumination pixels (about 1 million) in the wavefront method and to the number of sub-patches in the sub-patch method. The transillumination representation can potentially exploit the texture mapping hardware and can cache BRDF values, but without using these tricks, the sub-patch approach is significantly faster for scenes of less than a hundred thousand patches.

3.2. Refinement criterion

Having run the algorithm and computing the Monte-Carlo component as the difference of the reflected radiance and the finite-element component on the support of all basis functions, we can find where the Monte-Carlo component is large, thus the corresponding basis should be refined. Recall that in general the finite-element structure is formed by triangles with discretized hemispheres. Now the question is that

when the Monte-Carlo component over a triangle is significant, then whether the triangle or the directional hemisphere should be broken down.

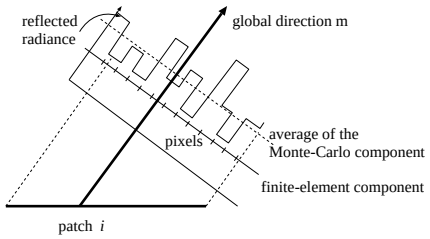


Figure 6: *Errors of the positional and directional discretization*

Since the input and output directions are the same for all points of the triangle, the error of the directional discretization appears as a constant shift for different pixels in the output buffer.

Considering this fact, let us use two measures:

$$V_s = \frac{1}{N}\sum(\Delta L_P - \frac{1}{N}\sum\Delta L_P)^2, \quad V_d = \frac{1}{N}(\sum\Delta L_P)^2$$

where P runs for the N different pixels covering the projection of the patches in the output buffer. These measures are averaged during the iteration. At the end of the phase if the average of V_s is high, then the corresponding triangle is decomposed into 4 sub-triangles. However, when V_d is high, then the directional hemisphere of the triangle is refined by dividing each discrete solid angle into four.

4. Optimizing further: incoming first shot

The proposed method selects bundles of parallel lines to transfer the radiance blindly without considering where the important sources are. On the one hand, this is good, since very many rays can be traced simultaneously in a single step (note that if the resolution of the transillumination buffer is 1000×1000, then a single transfer corresponds to tracing a million global lines where all intersections are used, i.e. at least 2 million rays). On the other hand, this becomes ineffective if the initial radiance is very heterogeneous due to the small bright light sources. These light sources need special treatment that is generally called as the *first-shot* or direct light source computation.

First-shot can be formulated as decomposing the radiance into emission and reflection and deriving an appropriate form of the rendering equation for the reflection term. Substituting the $L = L^e + L^r$ decomposition into the rendering equation, we can obtain the following formulae:

$$L^e + L^r = L^e + \mathcal{T}_{f_r}(L^e + L^r) \quad \Longrightarrow$$

$$L^r = (\mathcal{T}_{f_r}L^e) + \mathcal{T}_{f_r}L^r = L_{direct} + \mathcal{T}_{f_r}L^r,$$

This equation is similar to the original rendering equation. The only difference is that the original emission function L^e is replaced by its single reflection $L_{direct} = \mathcal{T}_{f_r}L^e$.

In order to compute and store the first reflection of the emission function, point samples are defined on the small light sources and hemicubes are placed above these point samples[25]. Running z-buffer/constant shading visibility algorithms for the sides of the hemicubes, the visible triangles and their visible portions can be identified. Then if the radiances of the grid points on a triangle point are computed, first it is determined whether or not the triangle is seen through some hemicube face from the light source. If it is seen, then the same texture mapping-like algorithm can be used to map pixels onto hemicube points. Another possibility is the calculation of those light vectors[29] on each patch, which can represent the illumination coming from the light sources. Note that unlike in other first-shot algorithms developed for diffuse radiosity, here the incoming radiance is stored, from which the outgoing radiance can be obtained by multiplying it with the BRDF taking into account the incoming and outgoing directions.

The combination of the proposed iteration with the incoming first-shot technique is quite straightforward. At a given iteration step not only the incoming radiance of previous transfer is reflected towards the new direction, but also the illumination of the light sources that are associated with the given patch.

5. Radiance updates during walk-through animation

In general animations both the objects and the camera may move. Walk-through animations represent an important special case when the objects are still but the camera may follow an arbitrary path. Walk-through involves a higher level of coherence among frames, thus more speed-ups can be expected from its proper utilization.

Let us now examine what happens if the eye position changes during the walkthrough animation. After first-shot, the decomposed radiance is

$$L = L^e + L_{direct} + \tilde{L}^r + \Delta L^r.$$

The projected reflected radiance $\tilde{L}^r$ is view independent thus remains valid for the new viewpoint. The emission function L^e should be re-evaluated at each sample point. The update of L_{direct} is also easy since the incoming radiance from the light source samples are stored. Using these incoming radiances and the BRDF function for the incoming directions and for the new view direction, the new values can be obtained quickly. The only term which poses difficulties is the Monte-Carlo component ΔL^r.

One way of handling this is to continue the stochastic iteration having altered the eye position. If the surfaces are not highly specular and the change of the view direction is small, then the sum of the emission, the direct reflection and

the finite-element approximation of the indirect reflection is a good approximation also for the next viewpoint, thus the iteration will converge quickly. If the sub-patch representation is used, then the Monte-Carlo component can also be reused in the next viewpoint. This requires an additional variable on each patch that stores the output radiance towards the eye due to the average Monte-Carlo component. When the view position changes, this value becomes an approximation, but it is usually better to start from this value than from zero. Clearly, starting from solution in the previous frame makes the errors of subsequent frames correlated. If the finite-element component is well adapted, the Monte-Carlo component represents just a small fraction of the total power in the scene, thus its inaccuracy does not create visible artifacts. The progressive nature of the algorithm and the fact that the error is correlated in different frames can be regarded as advantages in interactive applications. When the user moves quickly in the scene, although the computed image sequence becomes gradually inaccurate, but does not not exhibit flickering. When the user slows down at more interesting places, the algorithm has more time to refine the results, thus accurate images can be computed.

6. Simulation results

The presented algorithms have been implemented in C++ in OpenGL environment. The images have been rendered with 500×500 resolution. The transillumination buffer contained 1000×1000 pixels. The running times given in the following sections are measured on a laptop with 500 MHz Pentium III processor and with no graphics accelerator.

Concerning the refinement of the finite-element framework, practical experiences showed that it is usually not worth subdividing the directional sphere in scenes where the surfaces are not highly specular and therefore are the primary candidates for ray-bundle tracing. Thus it is enough to decide whether or not the surface triangles should be broken down.

Figure 7 shows a scene of a 3D Sierpinski set, that has 4479 patches. The diffuse albedo of the patches in this set is $(0.09, 0.03, 0.06)$ on the wavelengths 400 nm, 552 nm and on 700 nm, respectively. The specular albedo is wavelength independent and is between 0.8 and 0.4 depending on the viewing angle. The "shine" parameter of the max-Phong reflection model is 3. The walls are diffuse. The area lightsoure is sampled at 18 discrete points. The image was rendered with the wavefront method in 6 minutes. In addition to the rendered scene, the finite-element reflected component $\tilde{L}^r$ visualized by Gouraud shading and the direct illumination L_{direct} are also shown. Note that these two components really represent the major part of the radiance. On the other hand, the incorrect shadow smearing of the finite-element component in the ceiling is completely removed by the Monte-Carlo component in the final image.

The "Cornell chickens" scene of figure 8 was rendered with the sub-patch representation using 15082 sub-patches. The egg is purely specular and has albedo 0.9, the chickens have 0.5 specular albedo and about $0 \ldots 0.4$ wavelength dependent diffuse albedo, and the walls have $0 \ldots 0.6$ diffuse and 0.2 specular albedo. The shine parameter of all surfaces is 10. Note that non-diffuse global illumination of moderately complex scenes (15 K patches) becomes possible in about a minute.

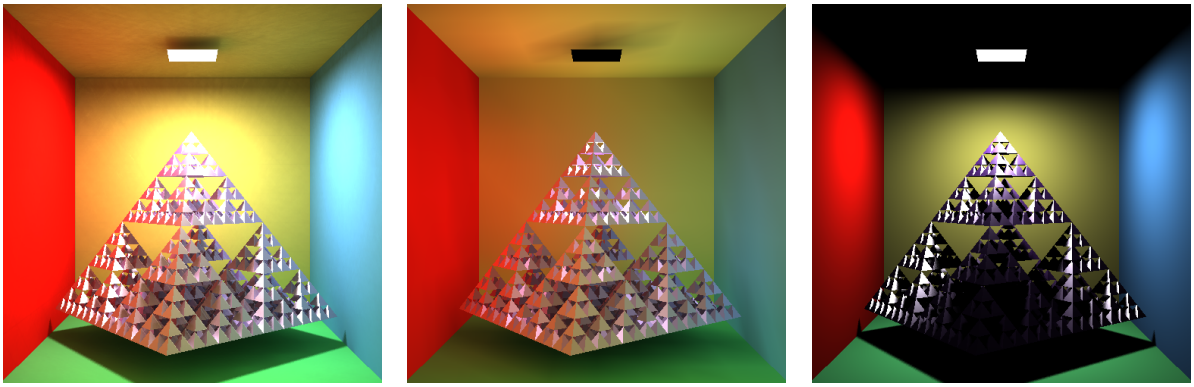
Figures 9 and 10 are snap-shots of an interactive walk-through in a virtual house. The model consists of 31.000 polygons that are tessellated to 241.000 sub-patches. A single iteration requires about 1 second. The user can expect a new frame in half a second, which is needed for the recalculation of the direct illumination, but in this case the glossy reflections of the indirect light become gradually inaccurate. Stopping for about 15 seconds, the algorithm quickly restores the accuracy.

7. Conclusions

This paper proposed the combination of continuous random walk and finite-element based iteration in a way that the asymptotic accuracy of the random walk and the high speed of iteration could be preserved. In addition to a general framework of combined algorithms, a particular method using ray-bundles has also been presented. Compared to previous ray-bundle tracing algorithms[22], the application of the combined method and the first-shot resulted in a speed-up factor of 10. This method applies a brute force strategy, which generates a large number of ray-paths exploiting coherence principles, but without taking into account local importance. Consequently, the method is particularly efficient for scenes of glossy reflection, and moderately complex models of tens of thousands of patches can be rendered in about a minute. If the solution is available, walk-through with close to interactive speed is possible. Due to the global nature of the algorithm, it cannot incorporate local importance sampling, thus it is strong for diffuse and glossy materials but weak for highly specular materials. The proposed method is worth being complemeted by bi-directional path-tracing algorithm governed by Metropolis sampling, which is particularly good for highly specular cases. The combination of the two methods is under development using the concepts of multi-pass rendering.

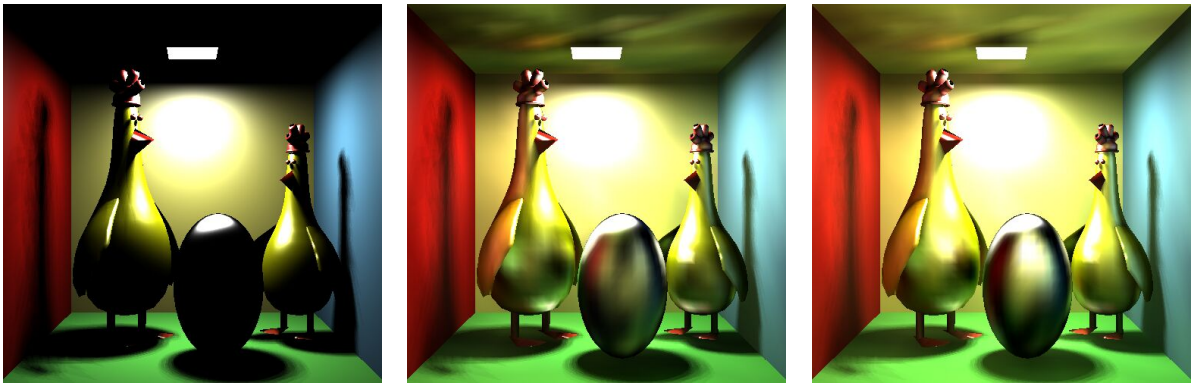
8. Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T029135). The architectural and Cornell-box scenes have been modelled by ArchiCAD and by Maya, respectively, that were generously donated by Graphisoft Ltd. and by Alias Wavefront.



L global illumination solution | $\tilde{L}^r$ finite-element component | L_{direct} direct illumination

Figure 7: *Sierpinski set rendered with the wavefront approach*



21 secs, first-shot | 36 secs, 20 iterations | 51 secs, 40 iterations

Figure 8: *The "Cornell chickens" after the first-shot and after 20, 40 iterations rendered with sub-patch representation*

Figure 9: *Shap-shots of a walk-through in a house modelled in ArchiCAD*

Figure 10: *A picture of the kitchen*

References

1. J. Arvo. Application of irradiance tensors to the simulation of non-lambertian phemomena. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 335–342, 1995.
2. Ph. Bekaert. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, University of Leuven, 1999.
3. 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.
4. M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 75–84, 1988.
5. Ph. 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.
6. 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.
7. H. W. Jensen. Importance driven path tracing using the photon maps. In *Rendering Techniques '95*, pages 326–335, 1995.
8. H. W. Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30, 1996.
9. J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 143–150, 1986.
10. 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.
11. A. Keller. Hierarchical Monte Carlo image synthesis. Technical Report 298/99, Universität Kaiserslautern, AG Numerische Algorithmen, 1999. to appear in Mathematics and Computers in Simulation.
12. E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, 1993.
13. L. Neumann. Monte Carlo radiosity. *Computing*, 55:23–42, 1995.
14. M. Sbert. *The Use of Global Directions to Compute Radiosity.* PhD thesis, Catalan Technical University, Barcelona, 1996.
15. P. Shirley. Time complexity of Monte-Carlo radiosity. In *Eurographics '91*, pages 459–466. Elsevier Science Publishers, 1991.
16. F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):187–198, 1991.
17. P. Slussalek. Photo-realistic rendering — recent trends and developments. In *Eurographics '97, STAR reports*, pages 35–57, 1997.
18. I. Sobol. *Die Monte-Carlo Methode.* Deutscher Verlag der Wissenschaften, 1991.
19. M. Stamminger, A. Scheel, A. Granier, F. Perez-Cazorla, G. Drettakis, and F. Sillion. Efficient glossy global illumination with interactive viewing. In *Graphics Interface'99, Kingston, Ontario*, 1999.
20. F. Suykens and Y. D. Willems. Weighted multipass methods for global illumination. *Computer Graphics Forum*, 18(3):209–220, 1999.
21. L. Szirmay-Kalos. *Monte-Carlo Methods in Global Illumination.* Institute of Computer Graphics, Vienna University of Technology, Vienna, 1999. http: //www.iit.bme.hu/˜szirmay.
22. L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum (Eurographics'99)*, 18(3):233–244, 1999.
23. L. Szirmay-Kalos. *Photorealistic Image Synthesis Using Ray-Bundles.* D.Sc. Dissertation, Hungarian Academy of Sciences, 2000. www.iit.bme.hu/˜szirmay/diss.html.
24. L. Szirmay-Kalos, B. Csébfalvi, and W. Purgathofer. Importance driven quasi-random walk solution of the rendering equation. *Computers and Graphics*, 23(2):203–212, 1999.
25. L. Szirmay-Kalos, M. Sbert, R. Martinez, and R.F. Tobler. Incoming first-shot for non-diffuse global illumination. In *Spring Conference of Computer Graphics '00*, 2000.
26. E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation.* PhD thesis, Stanford University, http://graphics.stanford.edu/papers/veach_thesis, 1997.
27. E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.
28. 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.
29. J. Zaninetti, P. Boy, and B. Peroche. An adaptive method for area light sources and daylight in ray tracing. *Computer Graphics Forum*, 18(3):139–150, 1999.