Unifying Continuous Random-Walk and Finite-Element Based Iteration Type Global Illumination Algorithms

László Szirmay-Kalos [*]
Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics
Budapest, Pázmány Péter s. 1/D, H-1112, HUNGARY
szirmay@iit.bme.hu

Abstract

The paper introduces a method that can combine 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. The classical iteration using finite-elements and random walks 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 walkthroughs and view-animation since when the viewpoint changes, only the small Monte-Carlo component needs to be recomputed. Using this approach quite complex scenes consisting of tens of thousands of surface elements can be rendered in about a minute, and when the solution is available, we can walk in the scene interactively.

Keywords: Global illumination, stochastic iteration, finite-element techniques, Monte-Carlo methods

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 and the reflection of all point radiances that are visible from here. The reflection of the visible points is expressed by an integral operator

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

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.

1. *Random walk* algorithms search light paths following a *depth-first* strategy [15, 9, 5, 12, 25]. 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 radiance of the point visible in this pixel, $\mathcal{T}_{f_r}$ is the light transport operator and L^e is the emission function. 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. 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 the $\mathcal{O}(m^{-(1-\epsilon)})$ convergence, which could be predicted for smooth integrands [21]. Since just discrete samples of the radiance and the geometry are needed, these methods can work with the original geometry and require no tesselation. The samples are generated independently, thus this approach is free from the error accumulation problem and can be run on parrallel machines easily. 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 utilize the space and time coherence of the radiance function. An exception is the Metropolis light transport [24] which obtains the new path by perturbating 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 the algorithms 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.

2. *Iteration* techniques, on the other hand, generate 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

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

thus 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 $\mathcal{T}_{f_r}$. 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 [21]. To store the radiance estimates, finite-element approaches should be used which represent the radiance function in a finite function series form:

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

Basis functions b_j are usually decomposed as 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 [18]. Due to the fact that the radiance has 4 variates and changes quickly, an accurate finite-element representation requires very many (a few million) 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. 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.

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. Furthermore, if the light-transport operator is not exactly evaluated, the limiting value is also distorted by the cumulative error. Thus only the initial behaviour 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 [26, 19] 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 [26], 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, 22]. A common problem of these methods is that the computational time given to a particular algorithm should be decided before starting the rendering and the different approaches cannot strenghten 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 solve the diffuse radiosity problem, the original problem is projected to a finite-element base, where it is solved by a special random walk. The speciality of this random walk is that in a single step several, but not all light-paths are advanced simultaneously.

2 The unified 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 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. 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 part seems to be similar to a classical variance reduction technique called the separation of the main part [17]. However, here the main part is not known in advance, neither is it analytically integrable. In [11] this problem has been solved by function approximation. Here the main part is generated adaptively and is integrated by a special Monte-Carlo quadrature rule.

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 [20]. 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:

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

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 [20].

3 Decomposition of the radiance function

Let us decompose the radiance function L to the emission L^e, to a reflected component $\tilde{L}$ that can be approximated by the linear composition 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:

$$L = L^e + \tilde{L} + \Delta L. \tag{3}$$

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.

4

Let us substitute this decomposition into the stochastic iteration formula:

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

The finite-element component is

$$\tilde{L}(m) = \sum_i \sum_j s_i d_j \cdot \langle L(m) - L^e, \tilde{s}_i \tilde{d}_j \rangle. \tag{4}$$

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

$$\tilde{L}(m) = \sum_i s_i(\vec{x}) \cdot \int_S \mathcal{T}_{\tilde{f}_r}^* L(m-1) \cdot \tilde{s}_i(\vec{x}) \, d\vec{x} \tag{5}$$

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

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

For example, if 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, then the resulting formulae are as follows:

$$\tilde{L}(m) = \sum_i s_i(\vec{x}) \cdot \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}_{\tilde{f}_r}^* L(m-1) \, d\vec{x} \tag{6}$$

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. Note that if a single directional basis function is used, then the projected BRDF becomes the albedo.

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

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

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

4   The new algorithm

The new unified 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:

$$\tilde{L}(m) = \frac{1}{m} \cdot \sum_{n=1}^{m} \tilde{L}'(n) = \frac{1}{m} \cdot \tilde{L}'(m) + \left(1 - \frac{1}{m}\right) \cdot \tilde{L}(m-1). \tag{7}$$

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 its behaviour 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 variances of the Monte-Carlo components for all basis function domains are evaluated. Then using these variances, the finite-element structure is refined and patches and solid angles are broken down appropriately.

The complete algorithm is:

UnifiedIteration
      $\tilde{L}(0) = 0, \Delta L(0) = 0$
      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
          $\Delta L(m) = L^r - \tilde{L}'(m)$
          $\tilde{L}(m) = 1/m \cdot \tilde{L}'(m) + (1 - 1/m) \cdot \tilde{L}(m-1)$
          $C(m) = 1/m \cdot \mathcal{M}(L^e + \tilde{L}(m) + \Delta L(m)) + (1 - 1/m) \cdot C(m-1)$
          Contribute to the variances $|\Delta L|_i$
          if m is an end of the phase
              Refine the finite-element structure based on the variances $|\Delta L|_i$
              Initialize the variances $|\Delta L|_i$ to zero
          endif
      endfor
  end

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, the following subproblems should be considered:

1. Temporary representation of the Monte-Carlo component $\Delta L(m)$ and the computation of its reflection to obtain the image estimate and the radiance in the next iteration cycle.

2. Refinement criterion.

5   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 which 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 light transport operator as required by equation 2:

$$E[\mathcal{T}_{f_r}^* L] = \int\limits_{\Omega'} 4\pi \cdot L(h(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta' \cdot \frac{1}{4\pi} \, d\omega' = \mathcal{T}_{f_r} L.$$

The radiance transfer needs the identification of the mutually visible points 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. Thus instead of considering all points, the visibility problem is discretized and is solved for the pixels. 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 results a pair of mutually visible points. For this pair of points, the radiance transfer is computed resulting in the reflected radiance L^r.

From the reflected radiance the finite-element component can be obtained by a simple averaging operation according to equation 6. Note that if the integral is evaluated on the window, then the cosine factor is compensated:

$$\tilde{L}(m)|_i = \frac{1}{A_i} \cdot \int\limits_{A_i} \mathcal{T}_{\tilde{f}_r}^* L(m-1) \, d\vec{x} \approx \frac{4\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P) \cdot \tilde{f}_r(\omega', P, \omega). \tag{8}$$

where P runs through 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.

Note that it would be straightforward to extend the method to take into account bi-directional transfers, however, we discuss only the one-directional formulation for the sake of notational simplicity.

Now let us consider the specific problems of the temporary representation of the Monte-Carlo component and the refinement criterion.

5.1   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 consequtive iteration. Since the radiance is transferred in a single direction through a discretized window, called transillumination buffer, snapshots of this discretized window can serve that purpose. Suppose that each pixel stores the radiance information arriving through the given pixel. 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 incoming radiance information. From the incoming radiance, the outgoing radiance for any possible direction can be obtained by multiplying with the local BRDF.

This 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. In both cases 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 distinquish from the transillumination buffer which serves as the input buffer. The correspondance of the coordinates of the input and output buffers is linear if parallel projection is used and rational for perspective projection. Note that the projection onto the transillumination buffers 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.

5.2   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 determine where
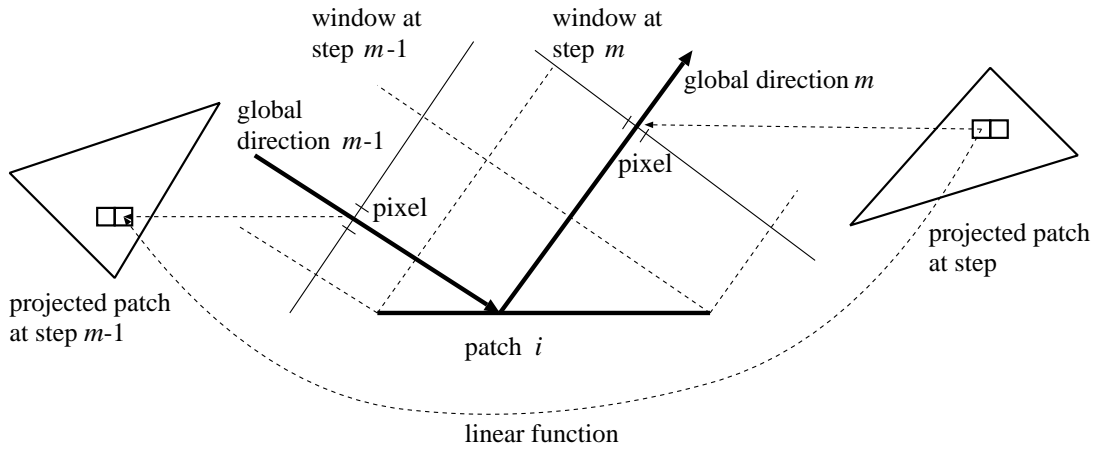
Figure 1: Reflection of the radiance wavefront of a ray-bundle

the Monte-Carlo component is large, thus the corresponding basis should be refined. Recall that 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.
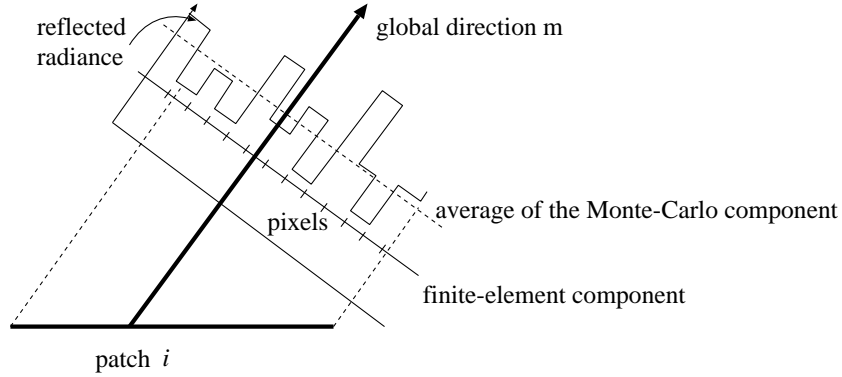

Figure 2: Errors of the positional and directional discretization

Since the input and output directions are the same for all points of the triangle, the error of the directional discretisation appears as a constant shift for different pixels in the output buffer. Considering this fact, let us use two variance 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 computed in every iteration step and averaged in each phase. Then at the end of the phase if the average of V_s is high, then the corresponding triangle is decomposed into 4 subtriangles. However, when V_d is high, then the directional hemisphere of the triangle is refined by dividing each discrete solid angle into four.

6   Optimizing further: incoming first shot

The proposed method selects bundles of parallel lines to transfer the radiance blindly without even 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 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 lightsources. These lightsources need special treatment that is generally called as the first-shot or direct lightsource 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 equation:

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

thus the obtained equation is similar to the original rendering equation. The only difference is that the original emission function is replaced by its single reflection.

In order to compute and store the first reflection of the emission function, point samples are defined on the small lightsources and hemicubes are placed above these point samples [23]. 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 lightsource. 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 [27] on each patch, which can represent the illumination coming from the lightsources. 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 this method with the proposed 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 lightsources that are associated with the given patch. Thus the overhead is just l BRDF computations per each patch at each iteration, where l is the number of those point lightsources and point samples on the area lightsources which are visible from the patch.

7 Radiance updates during the 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-throughs involve a higher level of coherence among frames, thus more speed-ups can be expected from its proper utilization.

So far we have assumed that the eye position and the objects are fixed. Let us now examine what happens if the eye position changes during the walkthrough animation. In the decomposed radiance

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

$\tilde{L}^r$ 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 lightsource 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 quickly obtained. 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. On the other hand, 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.
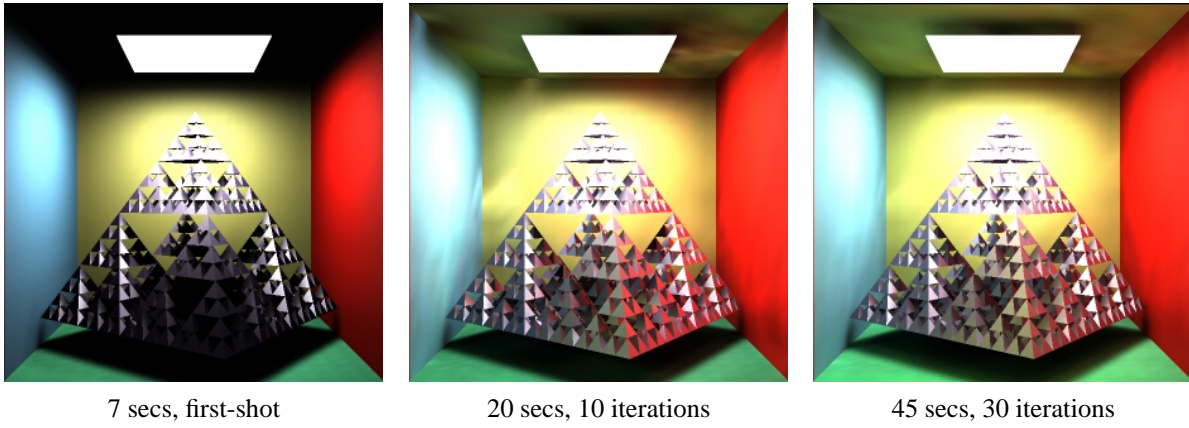
7 secs, first-shot   20 secs, 10 iterations   45 secs, 30 iterations

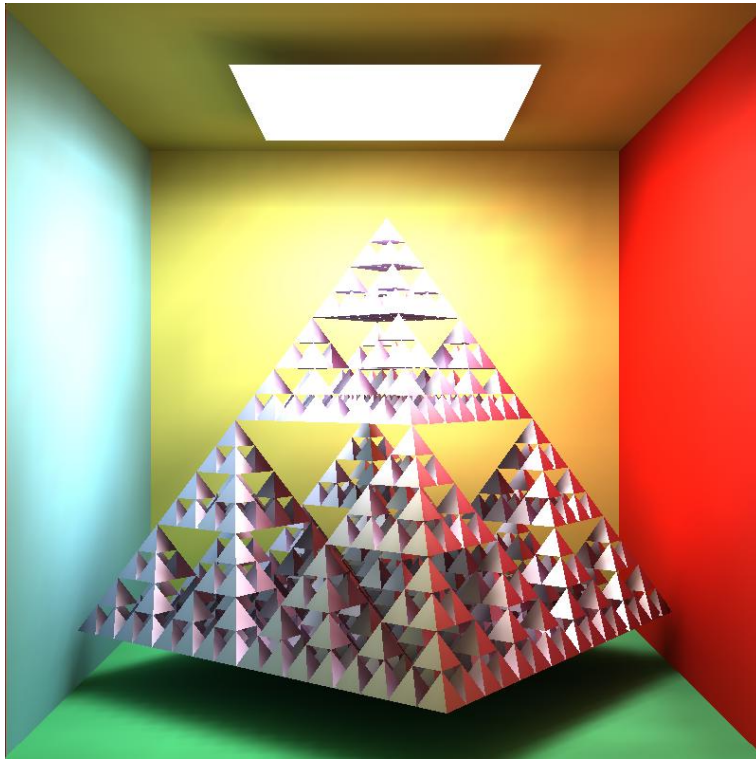Figure 3: Sierpiensky set after the first-shot and after 10 and 30 iterations
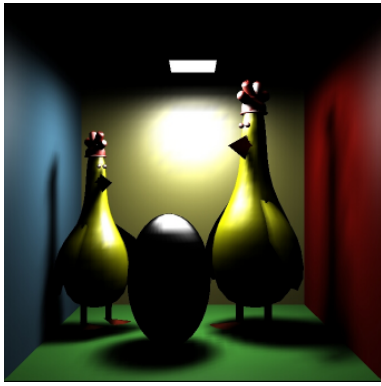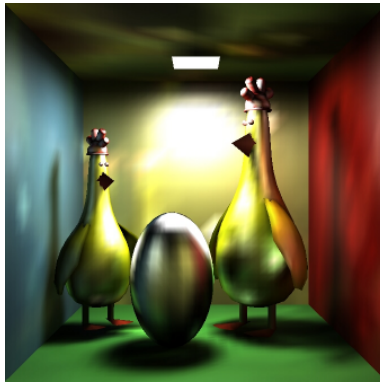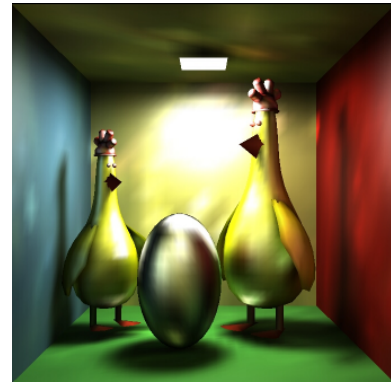

Figure 4: Sierpiensky set

| 7 secs, first-shot | 20 secs, 10 iterations | 45 secs, 30 iterations |

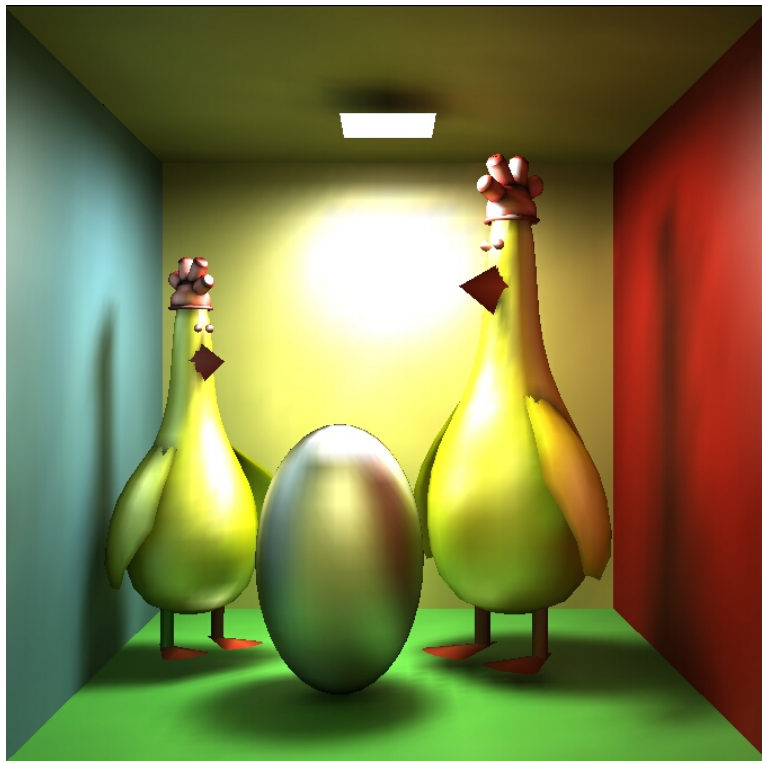Figure 5: The "Cornell chickens" after the first-shot and after 10, 30 iterations

Figure 6: The Cornell Chickens scene

8 Simulation results

The presented algorithms have been implemented in C++ in OpenGL environment. The images have been rendered with 800×800 resolution. The running times given in the following sections are measured on a laptop with 500 MHz Pentium III processor and with no graphics accelerator.

Figure 3 shows a scene with a 3D Sierpiensky set, that has 38552 patches. The diffuse albedo of the patches in this set is $(0.18, 0.06, 0.12)$ 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 is 3. The walls are diffuse.

In the "Cornell chickens" scene 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 $0 \ldots 0.4$ diffuse and 0.2 specular albedo. The shine parameter of all surfaces is 10.

Note that with the combination of the incoming first-shot and stochastic iteration, non-diffuse global illumination of moderately complex scenes (20 K patches) becomes possible in about a minute.

9 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 unified algorithms, a particular method using ray-bundles has also been presented. This method applies a brutal 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-throughs with close to interactive speed are possible.

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. http://www.cs.leuven.ac.be/ cwis/research/graphics/ CGRG.PUBLICATIONS/PHBPPHD.

[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. Verbal communication. 2000.

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

[18] 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.

[19] F. Suykens and Y. D. Willems. Weighted multipass methods for global illumination. *Computer Graphics Forum*, 18(3):209–220, 1999.

[20] L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum (Eurographics'99)*, 18(3):233–244, 1999.

[21] L. Szirmay-Kalos. *Photorealistic Image Synthesis Using Ray-Bundles*. D.Sc. Dissertation, Hungarian Academy of Sciences, 2000. www.iit.bme.hu/s̃zirmay/diss.html.

[22] 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.

[23] 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*, Budmerice, 2000.

[24] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, http://graphics.stanford.edu/papers/veach_thesis, 1997.

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

[26] 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.

[27] 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.