Eurographics Symposium on Rendering 2003
Per Christensen and Daniel Cohen-Or (Editors)

Global Illumination Animation with Random Radiance Representation

László Szirmay-Kalos, György Antal, Balázs Benedek

Department of Control Engineering and Information Technology, Technical University of Budapest
Budapest, Magyar Tudósok krt. 2., H-1117, HUNGARY
Email: szirmay@iit.bme.hu

Abstract

This paper proposes a non-diffuse global illumination algorithm that is fast enough to be appropriate for interactive walkthroughs and general animations. To meet the severe performance requirements, we heavily exploit coherence both in time and space, and use randomization to reduce the time and storage complexity. To speed up convergence and to support animation, the approximation of the radiance is stored in object space as well. However, in order to reduce the high memory requirements of such representations and to reduce finite-element artifacts, we use just a random approximation, which fluctuates around the real radiance function. The direction dependent radiance approximation is represented in a compact way, by four random variables per patch. The key of performance is then to make the error, i.e. the variance of this compact approximation as small as possible. In addition to main part separation, we apply a novel sampling scheme inspired by the Metropolis method to achieve this goal. In this algorithm light transfers are computed by both local and global methods using ray bundles and with the support of the graphics hardware. We conclude that both local and global approaches fail to efficiently compute all types of transfers, thus cannot be used alone. However, with the aid of multiple importance sampling, the merits of the two light transfer methods can be combined resulting in an algorithm that is robust and fast enough for animations. On the other hand, ray bundles, especially global ones, can update the illumination quickly when objects move, since they can efficiently identify which light paths became invalid.

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

1. Introduction

Animations can be classified as camera animations, also called walkthroughs[5], when only the camera moves, and as general animations when even objects are allowed to change their properties. Walkthroughs are simpler to compute since if we had the radiance function, they would only require us to identify the points visible from the new eye position and to obtain their radiance. However, the radiance is also a function of the viewing direction if the surfaces are non-diffuse, thus the explicit representation of this radiance function is usually not feasible [3]. General animations are even more difficult to render, since all properties, even light source intensity, may change in time.

Making global illumination fast enough to be appropriate for walkthrough and general animations is one of the most important challenges of rendering. To reach this goal, we can either try to increase the computation speed to a level that rendering from scratch takes just a fraction of a second, or we may exploit not only object space and view space coherence [9, 8] but also time coherence, and recompute only those parts of the illumination, which became invalid [7, 14, 30, 3, 4].

Taking into account the amount of computation required by the global illumination solution, the first approach is feasible only if we have huge computational power provided by a parallel system and/or we use simplifications [33, 15]. On the other hand, coherence allows interactive rendering even on a single computer. This paper proposes an algorithm that falls into this second category.

Coherence methods make the errors correlated. Sometimes it is an advantage since it can reduce dot noise and flickering. However, coherence can also have disadvantages, and can result in artifacts such as light leaks, for example. Due to time coherence the highlights and shadows may follow the movement of the objects with a noticeable delay. Such problems should be avoided by smart application of coherence, such as by good quality or adaptively subdivided meshes, continuous directional functions [25], and elegant heuristic strategies to locate discontinuities [33]. Concerning the problems of time coherence, we need a mechanism that quickly updates the changed illumination.

If we use random walks to transfer the light, we face the problem of slow convergence and of the task to figure out which walks are affected by these object movements. A brute force approach would regenerate all paths from scratch [33]. Alternatively, pioneer paths can also be selected to find the changes. Then the algorithm recomputes only those walks which are close to the pioneer paths reporting changes [13].

Adapting the solution to the continuously evolving environment is somehow natural in iteration approaches [2, 27]. In iteration the solution of the previous frame is supposed to be the initial value of the iteration, which will converge to the required solution with the speed of a geometric series. However, we should pay a high price for this remarkable convergence in terms of storage space, which becomes really prohibitive if the surfaces are non-diffuse, not to mention the visible artifacts of finite-element approximations.

To attack these problems, we propose an iteration algorithm with a novel random radiance approximation scheme that uses finite-element decomposition just in the spatial domain. The directional variation of the radiance is represented randomly, which requires just a few variables per patch, but provides a low variance estimate. Thus the proposed method is mesh based with continuous but random directional radiance representation. Due to the low variance random representation, the convergence rate of the iteration is preserved in the initial phase of the computation. The smaller random variations are eliminated by the slower Monte-Carlo quadrature computing the radiance only for the view directions of the patches.

Even the geometric convergence is too slow at parts of the scene where the radiance changes considerably during an animation sequence. Thus in our approach we follow a combined strategy, which is basically an iteration, but when objects move, it switches to a special mode, which removes previous transfers that have become invalid and introduces new ones as fast as possible.

2. Simulation of the light transport

In order to solve the global illumination problem, the light transport should be simulated. If the radiance estimate is represented by function $L(\vec{y},\omega')$, then the light transport produces a single reflection of the radiance function, which is obtained by applying operator $\mathcal{T}$:

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

where $\vec{y}$ is the point visible from $\vec{x}$ at direction $-\omega'$, Ω is the directional sphere, $f_r(\omega',\vec{x},\omega)$ is the bi-directional reflection/refraction function, and $\theta'_{\vec{x}}$ is the angle between the surface normal and direction $-\omega'$ at $\vec{x}$.

Since the light traverses the space along straight lines, the simulation requires the generation of lines to identify the points between the light is transported. There are many different possibilities for this line generation. Lines can be obtained deterministically or randomly as in Monte-Carlo algorithms. Monte-Carlo methods randomize the light transport operator, and apply a random operator $\mathcal{T}^*$ that gives back the effect of $\mathcal{T}$ in the average case:

$$E[\mathcal{T}^*L] = \mathcal{T}L.$$

Iterating a random transport operator makes the sequence not convergent. The radiance functions of the subsequent iteration steps will fluctuate around the real solution. However, the average of the iterated values will converge to the limiting value of the iteration of the original light transport operator [27]. It is not efficient to average the whole radiance function, since that would require its representation by finite-elements. Instead, averaging can be executed in image space, i.e. we compute the average of image estimates obtained from the fluctuating radiance sequence. Then pixel color $\mathcal{P}$ is obtained as an average of the estimates of all iteration steps

$$\mathcal{P}_n = \frac{1}{n}\cdot\sum_{k=1}^{n}\mathcal{M}L_k = \frac{1}{n}\cdot\mathcal{M}L_n + \left(1-\frac{1}{n}\right)\cdot\mathcal{P}_{n-1},$$

where $\mathcal{M}$ is the measuring operator associated with this pixel.

When iterating the original light transport operator, then the nth iteration step introduces the nth bounce of the light. A single stochastic iteration step, on the other hand, has two effects. In addition to generating the first estimate of the nth bounce, it also refines the estimates of all transfers of shorter lengths. Monte-Carlo algorithms take many samples and approximate the expected value as the average of these samples. It means that stochastic iteration requires much more iteration steps, especially if the variance caused by the random transport operator is high.

Random transport operators transfer the radiance between randomly sampled points connected by lines. The method may produce individual lines or a bundle of lines of certain similarity. Working with bundle of lines can exploit the coherence of the scene and can thus significantly increase the computation speed. The formation of bundles depends on what kind of similarity can be taken advantage of, and what kind of operations are supported by the hardware. For example, hemicube based radiosity algorithms consider lines with

the same origin and passing through a regular grid. The first intersection of these lines can be computed by the z-buffer hardware. Parallel ray-bundles can transfer the radiance of all points of the scene parallel to a random direction. The visibility needed by this parallel transfer can also be computed efficiently by incremental algorithms. Even if conventional ray-shooting is used, it is worth computing simultaneously those lines which visit the same nodes of the space partitioning data structure [17]. Realizing that current processors can execute four floating point instructions concurrently, it also seems advantageous to always follow four nearby lines [34].

Finally, line generation can also be classified according to the strategy of finding the starting point and its direction vector. *Local line methods* find the starting point of the half-line first, then they obtain the direction of the line. Tracing a ray into this direction will identify the other point of the transfer as the first intersection. An alternative is the *global line approach* [24, 22, 27] which samples the two points simultaneously.

There have been many discussions about the comparative advantages of the different algorithms, but no method can be claimed to be the best. This is not surprising since each method has advantages and disadvantages in certain situations. Thus instead of insisting on a given technique, it is worth combining several of them, in a way that the advantages are preserved. Such quasi-optimal combination of Monte-Carlo sampling techniques is offered by *multiple importance sampling* [31]. Suppose that we use different sampling methods, and sample z can be computed by method m with probability density $p_m(z)$. Assume also that method m is applied with probability P_m. Multiple importance sampling proposes to divide the integrand samples by the average probability, i.e. by

$$d(z) = \sum_m P_m \cdot p_m(z) \tag{1}$$

no matter which particular strategy generated the sample.

In our approach we use a combination of local and global bundles of rays (figure 1). The global method samples two interacting points simultaneously by a bundle of rays parallel to a random direction [22, 27]. The local method selects one point first, then partner points are sampled for the first point by a perspective ray-bundle. Although perspective ray-bundle transfer has proven to be effective in the hemicube method of deterministic diffuse radiosity, its application in Monte-Carlo algorithms is non trivial since it results in "corner spikes" (see section 2.2). In order to eliminate the drawbacks, a novel combination strategy is proposed that is based on multiple importance sampling.

In the next sections the elementary global and local methods are reviewed, then we discuss their combination.

2.1. Method 1: Parallel ray-bundle tracing

Parallel ray-bundle tracing transfers the radiance of all patches parallel to a randomly selected global line of direction ω' in each iteration cycle [27]. The random transport operator is:

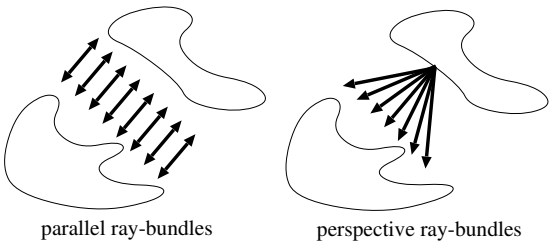


Figure 1: *Elementary methods to be combined*

$$\mathcal{T}_1^* L = 4\pi \cdot L(\vec{y}, \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \cos\theta'_{\vec{x}}.$$

where $\vec{y}$ is the point visible from $\vec{x}$ at direction $-\omega'$.

If the orientation is sampled uniformly, then its probability density is $p(\omega') = 1/4\pi$, thus the expectation of the random transport operator gives back the effect of the light transport operator $\mathcal{T}L$.

It is straightforward to extend the method to be bidirectional, which transfers the radiance not only into direction ω', but also to $-\omega'$. Note that this does not even require additional visibility computation. When working with bidirectional rays, the probability measure is changed to $1/2\pi$.

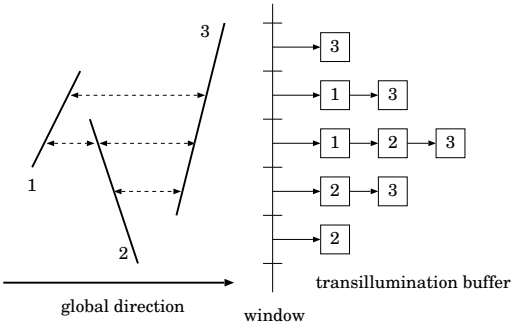


Figure 2: *Organization of the transillumination buffer*

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 (figure 2). The window is decomposed into a number of pixels. A pixel is capable of storing a list of patch indices and z-values. The lists are sorted according to the z-values. The collection of these pixels is called the *transillumination buffer*[22]. 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. By 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

average irradiance I caused by this transfer is stored at each patch:

$$I = \frac{2\pi \cdot \delta P}{A_i} \cdot \sum_P L^{in}(P),$$

where P runs on the pixels covering the projection of patch i, A_i is the area of this patch, $L^{in}(P)$ is the radiance of the surface point visible in pixel P in iteration step $n-1$, and δP is the area of the pixels. Note that the cosine term of the rendering equation is compensated by computing the integrand on the window instead of patch A_i. The radiance estimate in the direction of the eye is computed from the irradiance multiplying it with the BRDF taking into account the incoming and viewing directions. The average of these estimates results in the radiance in the direction of the eye on each patch. The final image is then rendered by Gouraud shading using the z-buffer hardware.

In the next iteration step, when the new direction is available, the irradiance is multiplied by the BRDF, resulting in a random estimate L^{rr} of the reflected radiance (superscript rr stands for the *random reflected* radiance). Currently, this algorithm is executed by the main processor, but the pixel shader implementation seems to be also feasible.

Parallel ray-bundle tracing samples point pairs independently of their radiance, making it possible to trace about two million rays in a single step. On the other hand, if the radiance distribution is heterogeneous, this can be ineffective, since most of these rays transfer just negligible illumination. This problem is inherent in all global approaches and is usually solved by applying a first shot [6, 29]. The first shot computes the reflection of small light sources, which are the primary causes of the inhomogeneous illumination, then replaces them by their first reflection. It means that parallel ray-bundles compute only the indirect illumination. However, in general animations when all objects and the light sources may move, this approach is not feasible since it would require the first shot step to be repeated in each frame. So we do not use the first shot here.

2.2. Method 2: Perspective ray-bundle shooting

Perspective ray-bundle shooting chooses a single patch randomly and sends its radiance from one of its randomly selected points towards all directions [1].

If patch j is selected with probability p_j and point $\vec{y}$ on this patch with uniform $1/A_j$ probability density, then the random transport operator is

$$(\mathcal{T}_2^* L)(\vec{x}, \omega) =$$

$$\frac{A_j}{p_j} \cdot v(\vec{x}, \vec{y}) \cdot L(\vec{y}, \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x} - \vec{y}|^2},$$

where ω' points from $\vec{y}$ to $\vec{x}$, and $v(\vec{x}, \vec{y})$ is the mutual visibility indicator, which is 1 if the two points are visible from each other and zero otherwise.

The points visible from $\vec{y}$ can be found by placing five window surfaces that form a hemicube around $\vec{y}$, and then using the z-buffer algorithm to identify the visible patches through these windows. Note that this is similar to the famous hemicube approach of the diffuse radiosity problem [10]. In fact, perspective ray-bundle shooting requires the vertex-patch form factors that can be computed by the hemicube.

As in parallel ray-bundles, the incoming radiance weighted by the point-to-point form factor are averaged on each patch, resulting a single irradiance value. In order to allow averaging the radiance values from different directions, we assume that the patches are small, and the directions towards the different points of the patch are approximately parallel. The radiance estimate from the camera is computed in the same way as discussed in the previous subsection. The only difference is that now the incoming directions are not the same on all patches, thus a direction value should also be stored on each patch.

According to importance sampling, it is worth setting the selection probability proportional to the integrand. Unfortunately, this is just approximately possible, and the patch selection probability is set proportional to the total power radiated by a given patch. If the light is transferred on several wavelengths simultaneously, the luminance of the radiated power should be used. Thus the selection probability of patch j is:

$$p_j = \frac{\mathcal{L}(\Phi_j)}{\sum_i \mathcal{L}(\Phi_i)},$$

where $\mathcal{L}$ is the luminance of a spectrum, and Φ_j is the spectral power of patch j. This way importance sampling can mimic the radiance of the source patch, but not the geometric factor between the source and the receiver. The geometric factor is inversely proportional to the square distance of the source and the receiver, thus is responsible for very high variation when the patches are close, for example around the corners (middle image of figure 3). This kind of "corner spikes" cause difficulties in many, otherwise very effective, shooting-type global illumination algorithms [33].

2.3. Combination of the ray-bundle based strategies

So far, we introduced two different random radiance transfer methods that use different sampling probabilities. Both of them are good for particular light transfers. Parallel ray-bundles are effective if the scene consists of patches of similar radiance, while perspective ray-bundles are good if one or several patches are much brighter than the others (note that these bright points are selected with much higher probability by perspective ray-bundle shooting). On the other hand, parallel bundles are accurate for transferring the radiance of close points, while perspective bundles for distant points. The reason is that parallel ray-bundle tracing uniformly samples the direction, and the probability that two surface elements see each other in a given direction decreases with their

distance. For perspective ray-bundles, the selection probability of two surface elements is independent of their distance. Thus a pair of close points is sampled by parallel ray-bundle tracing with higher probability than by perspective ray-bundle shooting. Thus dense scenes and corners can be rendered in a better way by parallel ray-bundle transfers.

Note that in figure 3 the image obtained with parallel bundles and without applying the first shot is generally worse than the image computed by perspective bundles, but parallel bundles do not introduce annoying bright spots at the corners.

In order to get the best of these two techniques, we combine them with the balance heuristic of multiple importance sampling, which requires the sampling densities of both techniques.

Parallel ray-bundle tracing samples the direction with a uniform density, i.e. the probability of generating a direction in $d\omega$ to find a partner point $\vec{y}$ from point $\vec{x}$ is

$$p_1(\omega) \cdot d\omega = \frac{d\omega}{2\pi}.$$

For perspective ray-bundle shooting, the probability that the shooting point is in differential area $d\vec{y}$ of patch j is

$$p_2(\vec{y}) \cdot d\vec{y} = \frac{\mathcal{L}(\Phi_j) \cdot d\vec{y}}{A_j \sum_i \mathcal{L}(\Phi_i)},$$

where Φ_j is the power of patch j and A_j is its area.

In order to apply the concept of multiple importance sampling, we have to solve the problem that different methods formulate the light transport with different integrals. Parallel ray-bundles use directional integrals while perspective ray-bundle shooting applies surface integrals. According to the formula of differential solid angles

$$d\omega = \frac{d\vec{y} \cdot \cos\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2},$$

directional integrals can also be converted to surface integrals, so the probability densities used by the discussed methods to sample a point $\vec{y}$ to shoot at point $\vec{x}$ are the following:

$$p_1(\vec{y}) = \frac{\cos\theta_{\vec{y}}}{2\pi \cdot |\vec{x}-\vec{y}|^2}, \quad p_2(\vec{y}) = \frac{\mathcal{L}(\Phi_j)}{A_j \sum_i \mathcal{L}(\Phi_i)}.$$

At each iteration step we decide randomly whether a perspective ray-bundle or a parallel ray-bundle will transfer the radiance of the scene, then the results are combined with multiple importance sampling. The probabilities of the two methods are P_1 and P_2, respectively ($P_1 + P_2 = 1$). According to equation 1 the divider of balanced heuristic becomes:

$$d(\vec{y}) = P_1 \cdot \frac{\cos\theta_{\vec{y}}}{2\pi|\vec{x}-\vec{y}|^2} + P_2 \cdot \frac{\mathcal{L}(\Phi_j)}{A_j \sum_i \mathcal{L}(\Phi_i)}.$$

When parallel ray-bundles are used, this weight should be multiplied by $d\vec{y}/d\omega = |\vec{x}-\vec{y}|^2/\cos\theta_{\vec{y}}$ in order to replace the density of surface points by the density of directions:

$$d(\omega) = P_1 \cdot \frac{1}{2\pi} + P_2 \cdot \frac{\mathcal{L}(\Phi_j)}{A_j \sum_i \mathcal{L}(\Phi_i)} \cdot \frac{|\vec{x}-\vec{y}|^2}{\cos\theta_{\vec{y}}}.$$

Let us interpret these results. When a perspective ray-bundle transfers the light in the combined method, the integrand of the rendering equation, i.e.

$$v(\vec{x},\vec{y}) \cdot L(\vec{y},\omega') \cdot f_r(\omega',\vec{x},\omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2},$$

is divided by $d(\vec{y})$ instead of its own sampling density $p_2(\vec{y})$. The integrand can be very large if the two points $\vec{x}$ and $\vec{y}$ are close, which is not compensated by the original density $p_2(\vec{y})$, resulting in high variance around the corners. However, thanks to parallel transfers, the combined density includes a similar $|\vec{x}-\vec{y}|^2$ factor, thus the corner spikes can be eliminated. On the other hand, when parallel bundles are used alone, the variance is caused by the variation of the source radiance. This error is also reduced in the combined method, since we divide the transfer by $d(\omega)$, which includes the source radiance thanks to the probability density of perspective transfers.

The optimal selection of P_1 and P_2 depends on how homogeneous the radiance is in the scene (we used $P_1 = P_2 = 0.5$ to render figure 3). It is worth setting the probability of perspective bundles high at the beginning of the algorithm and letting parallel ray-bundles refine the roughly distributed light energy. On the other hand, parallel ray-bundles force all patches to communicate, thus they can be efficiently used to detect changes during the animation.

In figure 3 the images computed with parallel (without first shot) and perspective ray-bundles can be compared with the result of the proposed combination method. Note that the combination algorithm can preserve the merits of both techniques and results in the most accurate image using the same computation time.

3. Random representation of the radiance

The discussed methods sample the radiance function in each step and obtain a new function. The radiance is a four variate function and usually has high variation. Our goal is to avoid the complete representation of this function, because that would pose prohibitive memory requirements. The surfaces are tessellated to patches, but the directional sphere is not decomposed to discrete solid angles. Instead we can store the irradiance of the last transfer (i.e. the incoming radiance estimate multiplied by the cosine of the incoming angle) and a direction on each patch. The irradiance is the average of the irradiances caused by the elementary rays hitting this patch in the given transfer, and the representative direction approximates the directions of the elementary rays (this approximation is exact in parallel transfers, but has a

Parallel Perspective Combined

Figure 3: *Comparison of stochastic iteration using parallel (left), perspective (middle) ray-bundles and the combination of the two methods (right) using the same computation time (5 secs)*

small error for perspective bundles, which is negligible if the patches are small).

For those patches that are not hit by rays, the irradiance of this iteration step is zero. From the irradiance I_n and incoming direction ω_n^{in} of iteration step n, the random approximation of the reflected radiance of the patch in an arbitrary direction ω can be obtained as

$$L_n^{rr}(\omega) = I_n \cdot f_r(\omega_n^{in}, \omega).$$

Examining the $L_n^{rr}(\omega)$ sequence, we can note that it has a high fluctuation since its elements are zero or very small when the patch is not the target of a transfer or the incoming direction is not the preferred direction of the BRDF, but when it is lucky enough to be hit by rays from the preferred direction, then it gets a larger contribution.

The variance of the whole method can be reduced if the fluctuation of this sequence is decreased. The general idea is to replace sequence I_n by another sequence, which is smoother but still results in the correct reflected radiance when averages are calculated. We use a combination of two techniques. The first is based on the *main part separation* [28, 19] and the second applies random acceptance and rejection according to Metropolis Sampling [21]. We should note that Metropolis sampling is used differently than in the Metropolis Light Transport algorithm [32]. Instead of sampling light paths proportional to their carried luminance, our objective is to develop a random representation of the directional radiance, which fluctuates around the real radiance. Metropolis sampling is used to control the samples in this fluctuating sequence, with the objective of keeping the luminance of the random radiance close to its average.

The first method separates the constant main part of the reflected radiance, which is replaced by its average. Let us store the directional average of the reflected radiance in variable L_n^d in each patch computed as

$$L_n^d = \frac{1}{n} \cdot \sum_{k=1}^{n} I_k \cdot \frac{a(\omega_k^{in})}{\pi} = \frac{1}{n} \cdot I_n \cdot \frac{a(\omega_n^{in})}{\pi} + \left(1 - \frac{1}{n}\right) \cdot L_{n-1}^d,$$

where $a(\omega)$ is the albedo of the material. Note that this main part is computed not only from the last transfer but from the average of all transfers that happened so far. We can take advantage of the fact that the main part is independent of the outgoing direction, and is valid for all directions. Thus a better (i.e. lower variance) sequence of the reflected radiance is

$$L_n^{rr}(\omega) = L_n^d + I_n \cdot \Delta f_r(\omega_n^{in}, \omega).$$

where Δf_r is the difference BRDF

$$\Delta f_r(\omega_n^{in}, \omega) = f_r(\omega_n^{in}, \omega) - \frac{a(\omega_n^{in})}{\pi}.$$

If we separated the BRDF to diffuse and specular terms instead of a/π and the difference BRDF, then the main part would be the diffuse reflection. This diffuse term, however, would be different from a radiosity solution since it would also incorporate the diffuse reflection of specular transfers.

The main part separation reduces the general fluctuation but the variation of the transfers represented by the difference BRDF still remains high in the sequence. Unfortunately, we cannot use the same trick of averaging here, since this term does depend on the outgoing direction ω, which will change from iteration cycle to iteration cycle. Either a finite element representation of the reflected radiance is needed, or we should store all incoming directions and irradiance values. Both approaches have prohibitive memory requirements.

The second variance reduction technique solves this problem without requiring additional variables. We shall still store a single incoming direction and irradiance per patch

in addition to the main part, but the incoming direction and the irradiance will not necessarily come from the last transfer (figure 4).

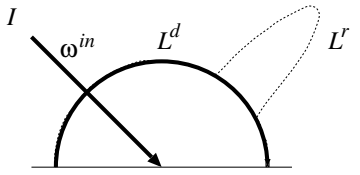


Figure 4: *Random representation of the radiance*

This method reduces the fluctuation by replacing a random sequence by another sequence of "similar samples". During the transformation zero samples are ignored, large samples of the original sequence will be scaled down and small samples will be scaled up. To measure the "size" of a sample, the luminance of the reflected irradiance is used. Since the reflection direction, i.e. the direction of the future transfers, is not known, a directional average is computed, which replaces the difference BRDF by its albedo $\Delta a(\omega)$. The "size" of a sample consisting of an irradiance value I and direction ω^{in} is then $\mathcal{L}(I\Delta a(\omega_n^{in}))$.

The transformation should not distort the expected values computed from the sequence, thus a scaled down value will appear more times in the new sequence. Scaling proportional to the "size" of the samples makes the luminance of the reflected radiance estimates similar:

$$\frac{I_n}{\mathcal{L}(I_n\Delta a(\omega_n^{in}))}\cdot C_n, \quad \text{where} \quad C_n = \frac{1}{n}\cdot\sum_{k=1}^{n}\mathcal{L}(I_k\Delta a(\omega_k^{in})).$$

The average computed from the transformed sequence will be correct if we can guarantee that I_m is expected to appear $\mathcal{L}(I_m\Delta a(\omega_m^{in}))/C_m$ times. A sampling scheme that can produce samples proportional to $\mathcal{L}(I_m\Delta a(\omega_m^{in}))$ is based on random acceptance and rejection similar to Metropolis sampling [18].

Suppose that before iteration step n, the irradiance and the incoming direction of an earlier step m are associated with a given patch. At each iteration step the new irradiance I_n is compared to the stored irradiance I_m. The new sample replaces the old one randomly, proportional to the ratio of their "sizes". If $\mathcal{L}(I_n\Delta a(\omega_n^{in}))$ is greater or equal than $\mathcal{L}(I_m\Delta a(\omega_m^{in}))$, then the new irradiance and its incoming direction will replace I_m and the stored incoming direction in the random representation of the radiance. However, when $\mathcal{L}(I_n\Delta a(\omega_n^{in}))$ is smaller than $\mathcal{L}(I_m\Delta a(\omega_m^{in}))$, the new irradiance is accepted randomly with probability $\mathcal{L}(I_n\Delta a(\omega_n^{in}))/\mathcal{L}(I_m\Delta a(\omega_m^{in}))$. According to the basic idea of Metropolis sampling this random acceptance happening with the ratio of the "sizes" results in a sequence of samples where the probability of obtaining a sample is proportional to its "size".

When combined with the separation of the main part, the improved sequence of reflected radiance estimates is

$$L_n^{rr}(\omega) = L_n^d + \frac{I_m\cdot\Delta f_r(\omega_m^{in},\omega)}{\mathcal{L}(I_m\Delta a(\omega_m^{in}))}\cdot C_n.$$

where I_m is the irradiance accepted most recently.

In order to establish importance sampling for perspective ray-bundles, the luminance of the patches should also be known. The computation of the powers from the irradiance values is also straightforward, the irradiance values should be multiplied by the albedos $a(\omega^{in})$ of the patches. The luminance of the power of a patch of area A is

$$\mathcal{L}(\Phi) = \Big(\mathcal{L}(L^e)\pi + \mathcal{L}(L^d)\pi + C_n\Big)\cdot A.$$

Finally, we emphasize that only the main part converges, but sequence $L_n^{rr}(\omega)$ will fluctuate around the main part forever. However, this does not pose any problem since the image is obtained as the average of the image estimates of subsequent iteration steps. Thus Monte-Carlo integration happens in image space, while we maintain a random, but low variance radiance estimate in object space. The final result will be the sum of the main part converging in object space and the average camera contributions of the fluctuating part, which converges in image space. The random radiance estimate stored in object space speeds up the iteration and supports animation as well.

4. Radiance updates in walkthrough animation

We proposed a random representation of the object space radiance. Since these values, including main part radiance L^d, irradiance I, incoming direction ω^{in} and scaling value C, are independent of the camera, they remain valid when the camera moves. When the camera moves, the new visible radiance values of the patches are set to

$$L^{eye}(\omega) = L^e(\omega) + L^d + \frac{I\cdot\Delta f_r(\omega^{in},\omega)}{\mathcal{L}(I\Delta a(\omega^{in}))}\cdot C.$$

This is a low variance estimator, especially if the surface is just moderately glossy or the source of the illumination is concentrated, thus even this initial value is quite close to the real visible radiance. Then, iterating further, a new image is computed as an average of the random estimates. Initial flickering can be reduced if the iteration is started from a weighted average of the previous and the new visible radiance values.

Figure 5 shows two displayed images and a temporary result of a walkthrough animation. This scene consists of 27 thousand patches having both diffuse and specular reflections. The wardrobe, which is the most specular object in this scene, has the following material properties: $(0.3, 0.3, 0.4)$ diffuse albedo on the wavelengths of R, G, B, 0.45 wavelength independent specular albedo, and the shininess of the Phong-like BRDF is 28. The probabilities of the parallel and

old camera position | after changing moving the camera | iterating further

Figure 5: *The left and right images show two frames of a walkthrough animation. The middle image is not seen by the user, but demonstrates the effect of just changing the camera location but not allowing time for the iteration to adapt to the new situation. Note that the algorithm needed a few iterations to correct the highlights.*

perspective transfers were 0.3 and 0.7, respectively. Note the slight difference between the middle image obtained after changing the camera and making a single iteration, and the right image taken after performing more iterations to get a converged image. This small difference shows that the proposed random radiance representation is quite accurate in glossy scenes. This approach allows 3 frames per second walkthrough on a 2GHz P4 computer if only three iterations are performed in each frame. Since the error in subsequent frames are highly correlated, the error due to the small iteration number is not noticeable for the user. Thanks to the spatial finite-element representation, there is no dot noise, and the frame rate is practically independent of the image resolution (we rendered the images at 800×800 resolution).

5. Radiance updates in general animation

In general animations objects may move and the emission of the light sources may change, which modifies the rendering equation. At the beginning of a frame the scene is represented by rendering equation $L = L^e + \mathcal{T}L$, and the approximation of its solution is available. Because of the changes of object properties, the new situation is described by a new light transfer operator $\mathcal{T}_{new}$ and a new emission function L^e_{new} in the next frame. The new radiance function L_{new} will be the solution of the updated rendering equation:

$$L_{new} = L^e_{new} + \mathcal{T}_{new} L_{new}.$$

Theoretically, we could continue the iteration with the new light transfer operator supposing the previous solution as the initial value, and the radiance will converge to the new solution. However, this is often not fast enough in animation sequences. Shadows may be visible in their old position for a few seconds. In order to avoid this, when objects move, we switch to a special iteration mode to quickly correct the radiance where it changed significantly.

Let us denote the difference of the new and the old radiance functions by $\Delta L = L_{new} - L$. Subtracting the old version of the rendering equation from the new one, we obtain:

$$\Delta L = \left(L^e_{new} - L^e + \mathcal{T}_{new} L - \mathcal{T}L\right) + \mathcal{T}_{new}\Delta L$$

We get an equation for ΔL, which is formally similar to the original rendering equation with the following light source term

$$L^{e*} = L^e_{new} - L^e + \mathcal{T}_{new} L - \mathcal{T}L.$$

It means that the same iteration algorithm can be continued to compute the change of the radiance function with this modified light source term. In order to work with the new light source term, the radiance transfer of each iteration cycle should be computed twice. First, placing objects at their original positions, the original radiance is transferred with negative sign (i.e. term $-\mathcal{T}L$ of L^{e*} is computed). Then, having moved the objects to their new positions, the new radiance is transferred with positive sign (i.e. $\mathcal{T}_{new}L + \mathcal{T}_{new}\Delta L = \mathcal{T}_{new}(L + \Delta L)$ is calculated).

These double transfers quickly update the illumination according to the new situation and after a few iterations, the shadows and highlights are moved to their new positions. At the end of this special iteration phase, the computed ΔL increments are added to the stored radiance representation (i.e. to the main part and to the scaling factor). In order to further refine the results, the algorithm switches back to the normal stochastic iteration scheme and iterates according to formula $L^e_{new} + \mathcal{T}_{new} L_{new}$.

Figure 6 shows two displayed images and temporary results of an object animation. The scene consists of 20 thou-

sand patches. The stripes of the egg have $(0.1, 0.2, 0.7)$ and $(0.8, 0.04, 0.04)$ diffuse albedos, 0.14 specular albedo, and the shininess values are 9 and 11, respectively. The rabbit's diffuse albedo is $(0.16, 0.19, 0.63)$, the specular albedo is 0.15, and the shininess is 9. The animation speed depends on the number of special iterations made to update the radiance. We have found that 10 iterations provide good images, which results in 1.5 frames per second. In interactive applications, however, users require a prompt response from the system, thus accuracy should be traded for speed. This is possible if the iteration number in the update cycles is reduced.

6. Conclusions

In this paper we proposed a random radiance representation scheme and an animation approach that can exploit both space and time coherence. This representation includes just a few values per patch, so the storage requirement is modest. The required storage is close to the need of a diffuse radiosity algorithm, although the proposed method is also good for glossy scenes. If the surfaces are not highly specular, the variance caused by the randomization is small due to the applied main part separation and the application of Metropolis sampling to maintain a constant luminance. Thus we can get the fast initial convergence of finite-element based iteration methods without their prohibitive memory requirements. The stochastic iteration algorithm used combines two random radiance transport methods based on multiple importance sampling. This novel combined strategy preserves the advantages of local and global light transfers, and eliminates the corner problem of local shooting and the necessity of the first shot of global sampling. The combined method is able to render moderately complex glossy scenes with the speed required by interactive systems. The application of parallel and perspective ray-bundles not only resulted in an effective global illumination algorithm, but proved to be really powerful to detect where the radiance function should be updated in an animation sequence.

Highly specular surfaces pose problems for this approach since they increase the variance of the random radiance representation and require higher tessellation levels to reconstruct the quickly changing radiance in the highlights. Fortunately, stochastic iteration applying bundles performs well on scenes containing a lot of patches until the rasterization and the radiance transfer through the pixels of the buffers are the bottlenecks of the computation, and not the geometric transformations. The rasterization time does not change if the patches are tessellated further, and the number of required iterations depends on the variation of the radiance function and not on the number of patches. On the other hand, the spatial finite-element representation eliminates the objectionable dot-noises, reduces the flickering of other Monte-Carlo algorithms and makes the algorithm practically independent of the image resolution.

7. Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T042735), the Bolyai Scholarship and the Slovene-Hungarian Fund. The architectural scenes have been modelled by ArchiCAD and by Maya, respectively, that were generously donated by Graphisoft Ltd. and by AliasWavefront.

References

1. Gy. Antal, L. Szirmay-Kalos, F. Csonka, and Cs. Kelemen. Multiple strategy stochastic iteration for architectural walkthroughs. *Computers & Graphics*, 27:285–292, 2003.
2. Ph. Bekaert. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, University of Leuven, 1999.
3. G. Besuievsky and M. Sbert. The multi-frame lighting method - a Monte-Carlo based solution for radiosity in dynamic environments. In *Rendering Techniques '96*, pages pp 185–194, 1996.
4. Gonzalo Besuievsky and Xavier Pueyo. A monte carlo method for accelerating the computation of animated radiosity sequences. In *Proceedings of Computer Graphics International 2001*, pages 201–208, 2001.
5. Kadi Bouatouch and Sumanta N. Pattanaik. Interactive Walkthrough Using Particle Tracing. In Rae E. Earnshaw and John A. Vince, editors, *Computer Graphics Developments in Virtual Environments (CG International '95 Proceedings)*, Boston, MA, 1995. Academic Press.
6. F. Castro, R. Martinez, and M. Sbert. Quasi Monte-Carlo and extended first-shot improvements to the multi-path method. In *Spring Conference on Computer Graphics '98*, pages 91–102, 1998.
7. Shenchang Eric Chen. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 135–144, August 1990.
8. P. Christensen. Faster photon map global illumination. *Journal of Graphics Tools*, 4(3):1–10, 2000.
9. 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.
10. M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH '85 Proceedings)*, pages 31–40, 1985.
11. C. Damez and F. Sillion. Space-time hierarchical radiosity. In *Rendering Techniques '99*, pages 235–246, New York, NY, 1999. Springer Wien.

12. Cyrille Damez, Kirill Dmitriev, and Karl Myszkowski. Global illumination for interactive applications and high-quality animations. Eurographics, September 2002. STAR - State of the Art Report.

13. Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Rendering Techniques 2002 (Proceedings of the Thirteenth Eurographics Workshop on Rendering)*, June 2002.

14. George Drettakis and Francois X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 57–64, 1997.

15. Dieter Fellner, Stephan Schaefer, and Marco Zens. *Parallel Computing: Fundamentals, Applications and New Directions*, volume 12 of *Advances in Parallel Computing*, chapter Photorealistic Rendering in Heterogeneous Networks. Elsevier Science, 1998. Proceedings of Parallel Computing '97.

16. X. Granier and G. Drettakis. Incremental updates for rapid glossy global illumination. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, volume 20, pages C–268–C–277, September 2001.

17. V. Havran. *Heuristic Ray Shooting Algorithms*. Czech Technical University, Ph.D. dissertation, 2001.

18. Cs. Kelemen, B. Benedek, and L. Szirmay-Kalos. Bidirectional rays in global illumination. In *WSCG 2003 Conference, Posters, Plzen*, 2003.

19. 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.

20. Igancio Martin, Xavier Pueyo, and Dani Tost. Frame-to-frame coherent animation with two-pass radiosity. Technical Report IIiA 99-08-RR, Institut d'Informatica i Aplicacions, Universitat de Girona, Girona, Spain, June 1999.

21. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

22. L. Neumann. Monte Carlo radiosity. *Computing*, 55:23–42, 1995.

23. J. Nimeroff, J. Dorsey, and H. Rushmeier. Implementation and analysis of a global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4), 1996.

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

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

26. Marc Stamminger, Annette Scheel, Xavier Granier, Frederic Perez-Cazorla, George Drettakis, and Francois Sillion. Efficient glossy global illumination with interactive viewing. *Computer Graphics Forum*, 19(1):13–25, 2000.

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

28. L. Szirmay-Kalos, F. Csonka, and Gy. Antal. Global illumination as a combination of continuous random walk and finite-element based iteration. *Computer Graphics Forum (Eurographics'2001)*, 20(3):288–298, 2001.

29. 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.

30. Parag Tole, Fabio Pellicini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002 Annual Conference)*, 21(3):537–546, 2002.

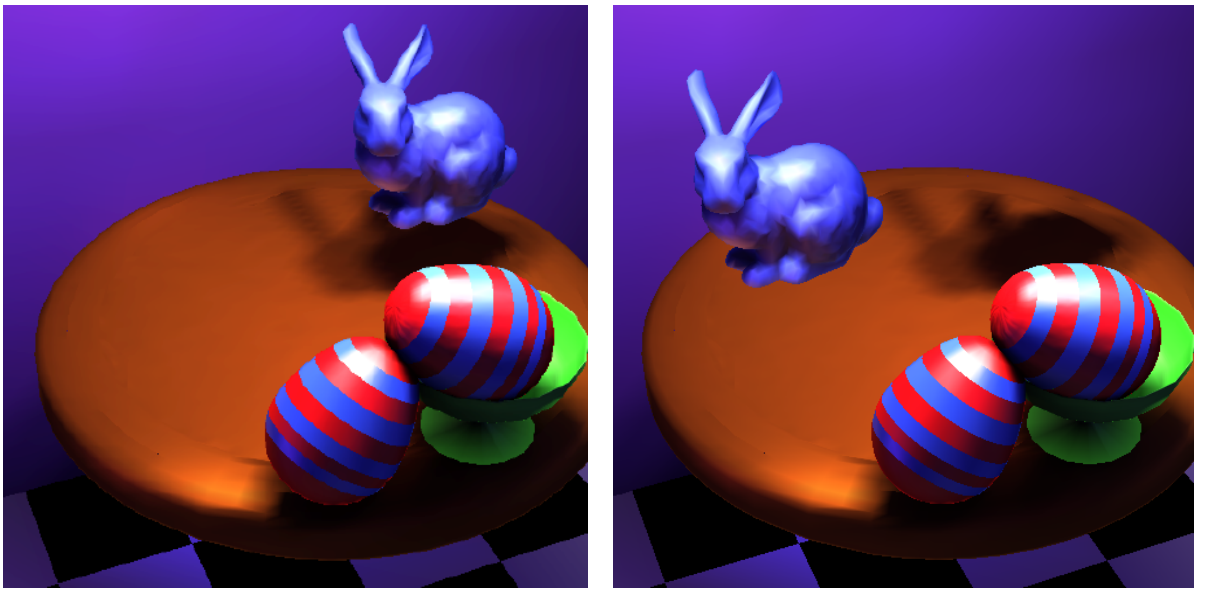
31. E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, http://graphics.stanford.edu/papers/veach_thesis, 1997.

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

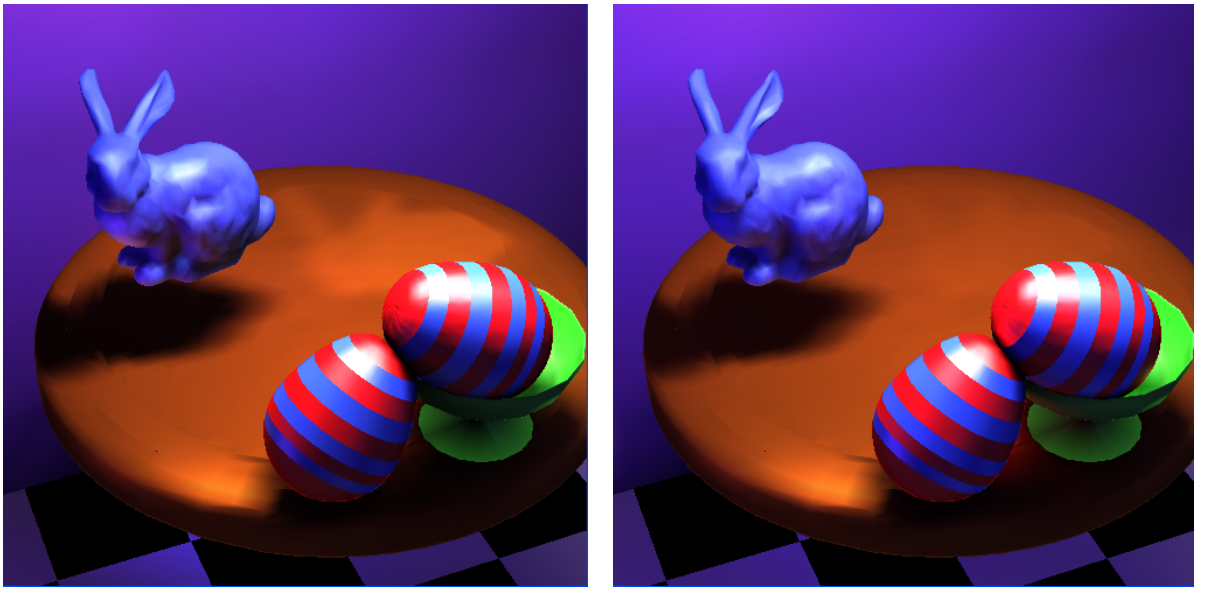
33. I. Wald, C. Benthin, and P. Slussalek. Interactive global illumination using fast ray tracing. In *13th Eurographics Workshop on Rendering*, 2002.

34. I. Wald, C. Benthin, P. Slussalek, and M. Wagner. Interactive rendering with coherent ray tracing. In *Eurographics '01*, 2001.

35. 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.



old | after moving the object



end of adaptation | continuing the iteration

Figure 6: *The first and last images show two frames of an object animation rendered at 1 frame per second on a Pentium 4, 2GHz computer. The two other images are not seen by the user, but demonstrate the roles of the adaptation phase.*