Multiple Strategy Stochastic Iteration for Architectural Walkthroughs

György Antal, László Szirmay-Kalos, Ferenc Csonka, Csaba Kelemen

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
Architectural walkthroughs require fast global illumination algorithms and also accurate results from certain viewpoints. This paper introduces a global illumination method that combines several strategies to meet the contradicting criteria of architectural walkthroughs. The methods include parallel and perspective ray-bundle shooting and ray shooting. Each method is designed to randomly approximate the effect of the light transport operator. Parallel ray-bundle tracing transfers the radiance of all points parallel to a randomly selected global direction, with perspective ray-bundles we can shoot the radiance of a single patch in all directions, and ray shooting transfers the radiance of a randomly selected point at a randomly selected direction. These strategies are of complementary character since each of them is effective in different illumination conditions. The proposed algorithm is iterative and the steps realized by different methods that randomly follow each other. In each step, the applied strategy is selected randomly according to the properties of the current radiance distribution, thus we can exploit that the used strategies are good in different conditions. The formal framework of their combination is the stochastic iteration. Although the final result is the image, i.e. the algorithm is view dependent, a rough approximation of the radiance function is stored in object space, that can allow fast movements at reasonable storage requirements and also speed up Monte-Carlo simulations which result in the final image. The method is also suited for interactive walkthrough animation in glossy scenes since when the viewpoint changes, the object space radiance values remain valid and the image quickly adapts to the new situation.

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

1. Introduction

In architectural CAD programs, the designer expects physically correct lighting when he walks through the virtual building. The rendering algorithm should be fast enough to allow interactive movements. Fortunately, the computed illumination should not be very precise when the designer walks quickly, but when he stops to carefully examine a certain region, the image should get more and more accurate.

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 computa-

presented at Dagstuhl Seminar on Monte-Carlo Methods in Rendering 2001.

tion 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[9, 6, 4, 7, 17]. 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 which estimate the integrals using m random samples. Since Monte-Carlo methods have $\mathcal{O}(m^{-0.5})$ convergence independently of the dimension of the integration domain, they can avoid the exponential core of classical quadrature rules[13]. 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[3, 1]. 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:

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

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[15]. 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[5], spherical harmonics[10] or Haar functions[12]. 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 shadows and highlights may be placed incorrectly[11]. 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.

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. In our targeted application area, in architectural walkthroughs, images should be generated usually close to real-time and but when we stop to look at small details, we have the time to wait for more accurate images. Considering the better initial convergence and the view independence, iteration seems to be the better alternative. In order to allow the temporary representation of the radiance function, the surfaces are decomposed to small patches that can be assumed to have homogeneous points. Thus the radiance of each surface is described by a directional function, which can be obtained as the average of the directional radiance of the points of the patch. If deterministic iteration were used, this directional radiance function should be approximated by many directional basis functions, which could easily lead to intorelable memory requirements. Thus we choose a different approach istead.

In order to reduce its astronomical storage requirements and to solve the error accumulation problem, iteration is randomized, which leads to stochastic iteration. The formal basis is the stochastic iteration [14], which replaces the light-transport operator by a random transport operator that gives back the effect of 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

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

This version does not provide converged radiance values in the object space, just in the image space. This is an advantage since we could get rid of the storage requirements of the direction dependent radiance values on each patches. However, it is worth separating the main part of the radiance function that is simple to represent and store it explicitely. This has two benefits. On the one hand, we can use it to reduce the variance of the Monte-Carlo simulation. On the other hand, this main part represents the radiance in a view independent way, which can be taken advantage of in fast interactive walkthrough.

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[16]:

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

In order keep the storage low, we allow just one directional basis function per patch to represent the final element part (the extension to higher elements is straightforward). In this case the finite element part can be obtained as a simple average of the directional radiance function of the patch:

$$\tilde{L} = \frac{1}{\pi} \cdot \int_{\Omega} L(\omega) \cdot \cos\theta \, d\omega.$$

If the patch receives illumination just from direction ω' and the irradiance is $I(\omega')$ — this will be the usual case in the algorithm — then the average radiance can be obtained from the albedo:

$$\tilde{L} = I(\omega') \cdot \frac{1}{\pi} \cdot \int_{\Omega_j} f_r(\omega', \vec{x}, \omega) \cdot \cos\theta \, d\omega = I(\omega') \cdot \frac{a(\vec{x}, \omega')}{\pi}.$$

The albedo can be computed in the preprocessing phase for each possible material and stored in tables or can be estimated on the fly.

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

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

To obtain the radiance value of patch i, the radiances of its points are averaged:

$$L(m)|_i = \frac{1}{A_i} \cdot \int_{A_i} \mathcal{T}^*_{f_r} L(m-1) \, d\vec{x} \qquad (4)$$

In each iteration step the radiance average is obtained an image estimate is computed from the actual radiance. Note that the image estimates and the finite-element components obtained in an iteration step, 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 finite-element components of different iteration steps. 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.

The complete algorithm is:

```
StohasticIteration
    $\tilde{L}(0) = 0$, $\Delta L(0) = 0$
    for $m = 1$ to $M$ do
        $L^r = \mathcal{T}^*_{f_r}(L^e + \tilde{L}(m-1) + \Delta L(m-1))$
        $\tilde{L}'(m)$ = average of $L^r$
        $\Delta L(m) = L^r - \tilde{L}'(m)$
        $\tilde{L}(m) = 1/m \cdot \tilde{L}'(m) + (1 - 1/m) \cdot \tilde{L}(m-1)$
        $C'(m) = \mathcal{M}(L^e + \tilde{L}(m) + \Delta L(m))$
        $C(m) = 1/m \cdot C'(m) + (1 - 1/m) \cdot C(m-1)$
    endfor
    Display $C(m)$ colors
end
```

The dataflow of the new algorithm is shown in figure 1. 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.

This is a generic algorithm from which different specific versions can be built by inserting the random transport operator. The algorithm will be fast if the application of the

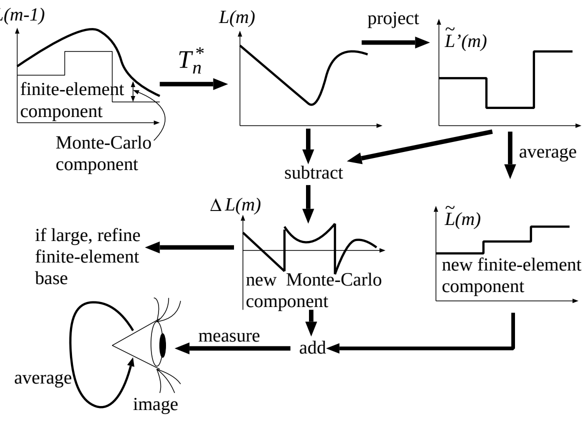

Figure 1: *Dataflow in the new algorithm*

random transport operator $\mathcal{T}^* L$ results in a low variance random variable. Note that this depends not only on the random transport operator but also on the current radiance function. With other words, for a different actual radiance function, different transport operator can be the winner of this game.

Now, let us consider three candidate methods to realize a single step of the stochastic iteration.

2. Parallel ray-bundle tracing

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

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

Indeed, if the direction 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 $\mathcal{T}_{f_r} L$, as required by equation (2):

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

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*[8]. 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.

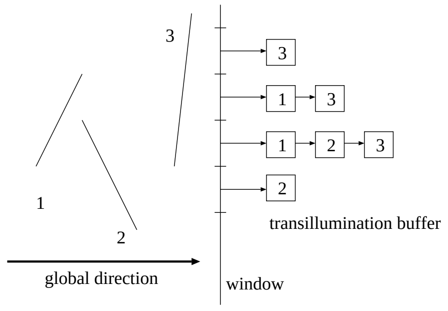

Figure 2: *Organization of the transillumination buffer*

From the reflected radiance the patch radiance 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:

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

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

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.

2.1. Perspective ray-bundle shooting

Perspective ray-bundle shooting selects a single patch randomly and sends its radiance from one of its randomly selected point towards all directions. According to importance sampling, it is worth setting the selection probability p_i proportional to the powers of the patches.

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

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

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

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

The expected value of this random variable is:

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

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

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

Using the formula of solid angles $d\vec{y} \cdot \cos\theta_{\vec{y}}/|\vec{x}-\vec{y}|^2 = d\omega_{\vec{x}}$ and assuming that illumination can only come from surfaces — i.e. there is no external sky light illumination — the integration over all surfaces can be replaced by an integration over all incoming solid angles:

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

and this is exactly what we wanted to prove.

To obtain the patch radiance, the radiances of the points are averaged:

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

$$\frac{A_j}{p_j A_i} \int_{A_i} v(\vec{x},\vec{y}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{x}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \frac{\cos\theta'_{\vec{x}} \cdot \cos\theta_{\vec{y}}}{|\vec{x}-\vec{y}|^2}\, d\vec{x} \tag{6}$$

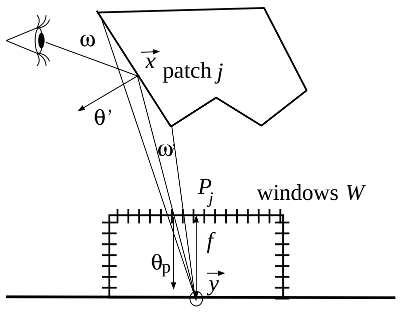

Figure 3: *Perspective ray-bundle tracing with hemicubes*

The integral in equation (6) can also be evaluated on the five window surfaces (W) that form a hemicube around the source $\vec{y}$ (figure 3). Note that this is similar to the famous hemicube approach of the diffuse radiosity problem [2]. In fact, radiance shooting requires the vertex-patch form factors that can be computed by the hemicube. In this section, we re-derive the basic formulae to show that they can also be used in cases when the reflection is non-diffuse.

To find formal expressions, let us express the solid angle $d\Omega_p$, in which a differential surface area $d\vec{x}$ is seen through pixel area $d\vec{p}$, both from the surface area and from the pixel area:

$$d\Omega_p = \frac{d\vec{x} \cdot \cos\theta'_{\vec{x}}}{|\vec{y}-\vec{x}|^2} = \frac{d\vec{p} \cdot \cos\theta_p}{|\vec{y}-\vec{p}|^2}, \tag{7}$$

where θ_p is the angle between direction pointing to $\vec{x}$ from $\vec{y}$ and the normal of the window (figure 3). The distance $|\vec{y}-\vec{p}|$ between pixel point $\vec{p}$ and the lightsource $\vec{y}$ equals to $f/\cos\theta_p$ where f is the distance from $\vec{y}$ to the window plane, that is also called the *focal distance*. Using this and equation (7), differential area $d\vec{x}$ can be expressed and substituted into equation (6), thus we can obtain:

$$L(m)|_i =$$

$$\frac{A_j}{p_j A_i f^2} \cdot \int_W v(\vec{y},\vec{x}) \cdot L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}, \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos^3\theta_p\, d\vec{p}.$$

Let P_i be the set of those pixels in which patch i is visible from the lightsource. P_i is computed by running a z-buffer/constant shading rendering step for each sides of the window surface, assuming that the color of patch i is i, then reading back the "images". The reflected radiance on patch i is approximated by a discrete sum as follows:

$$L(m)|_i \approx$$

$$\frac{A_j \delta P}{p_j A_i f^2} \cdot \sum_P L(\vec{y}, \omega'_{\vec{y}\to\vec{p}}) \cdot f_r(\omega'_{\vec{y}\to\vec{x}}, \vec{x}(\vec{p}), \omega) \cdot \cos\theta_{\vec{y}} \cdot \cos^3\theta_p,$$

where δP is the area of a single pixel in the image. If R is the resolution of the image — i.e. the top of the hemicube contains $R \times R$ pixels, while the side faces contain $R \times R/2$ pixels – then $\delta P = 4f^2/R^2$.

2.2. Ray shooting

In this method the random transport operator uses independent rays having random origin $\vec{y}_i$ and direction ω_i generated with a probability density $p(\vec{y},\omega)$ that is preferably proportional to the cosine weighted radiance of this point at the given direction. This ray transports the power

$$\Phi(\vec{y},\omega') = \frac{L(\vec{y},\omega')\cos\theta_{\vec{y}}}{p(\vec{y},\omega')}$$

to that point $\vec{x}$ which is hit by the ray, where it is reflected, modifying the radiance function. On a single wavelength, the probability of reflection is the BRDF times the cosine of the outgoing angle, i.e.

$$f_r(\omega', \vec{x}, \omega) \cdot \cos\theta_{\vec{x}},$$

but the cosine angle is compensated when the power is converted to radiance. Formally, the random transport operator is

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

$$\frac{L(\vec{y},\omega')\cos\theta_{\vec{y}}}{p(\vec{y},\omega')} \cdot \delta(\vec{x} - h(\vec{y},\omega')) \cdot f_r(\omega', \vec{x}, \omega), \tag{8}$$

where δ is the Dirac delta function that expresses that now only that point can receive radiance which is hit by the ray. Using the definition of the solid angle, $d\omega_{\vec{y}} = d\vec{x} \cdot \cos\theta'_{\vec{x}} / |\vec{y} - \vec{x}|^2$, a symmetry relation can be established

$$d\vec{y} \cdot d\omega_{\vec{y}} \cdot \cos\theta_{\vec{y}} = d\vec{y} \cdot \frac{d\vec{x} \cdot \cos\theta'_{\vec{x}}}{|\vec{y} - \vec{x}|^2} \cdot \cos\theta_{\vec{y}} =$$

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

which allows us to easily prove that the requirement of equation (2) holds, that is, the expectation of the random transport operator defined in equation (8) really gives back the original light transport operator:

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

$$\int_S \int_{\Omega'_{\vec{y}}} \frac{L(\vec{y}, \omega') \cos\theta_{\vec{y}}}{p(\vec{y}, \omega')} \cdot \delta(\vec{x} - h(\vec{y}, \omega')) \cdot f_r(\omega', \vec{x}, \omega) \cdot p(\vec{y}, \omega')\, d\vec{y} d\omega' =$$

$$\int_{\Omega'_{\vec{x}}} L(h(\vec{x}, -\omega'), \omega') \cdot f_r(\omega', \vec{x}, \omega) \cdot d\omega'_{\vec{x}}.$$

According to importance sampling, $p(\vec{y}, \omega)$ is preferably proportional to radiance of the point $\vec{y}$ at direction ω. This sampling can be realized in two steps. First the patch is selected with a probability proportional to its power. Then $\vec{y}$ is found with a uniform distribution on the selected patch.

3. Representation of the temporary radiance

All the three 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, thus its accurate finite-element representation would require many basis functions. Fortunately, the necessary storage space can be greatly reduced if the complete evaluation of the new radiance function is postponed until the new evaluation point and direction are already known. Note that all the three methods require just partial information about the radiance function, parallel ray-bundle transfer needs the radiance values just in a single direction, perspective ray-bundle transfer requires the radiance distribution of a single patch only, while ray shooting uses the radiance of single point and at a single direction. In an iteration step let us thus compute only the irradiance on each patch, which is independent of the transfer direction of the next step. With the irradiance information we also store the incoming direction. In the next iteration step, when the output radiance of a patch in a given direction is needed, then it is obtained on the fly, multiplying the irradiance by the BRDF of the patch taking into account the previous and current directions.

In order to establish importance sampling for the perspective and ray shooting transfers, the powers of the patches should also be known. The computation of the powers from the irradiance values is also straightforward, the irradiances should be multiplied by the albedos $a_i(\omega)$ of the patches.

4. Multiple strategy algorithm

So far, we introduced three different random radiance transfer methods. Each of them is good for a particular radiance distribution. The parallel ray-bundles are effective if the scene consists of patches of similar radiance, while the perspective ray bundles and ray shooting are effective if one or several patches are much brighter than the others. In this case, we can prefer ray-shooting to perspective ray-bundle transfer if the patch is highly specular.

Note that in random walk algorithms it is not a good idea to alter the sampling according to the currently transferred radiance since that can make the method biased, but in stochastic iteration this poses no problems. We conluded that all three methods meet the requirement of stochastic iteration, i.e. the expected value of their application gives back that of the real light transport operator. Obviously, any random combination of the three methods, where the probabilities sum up to 1, leads also a valid stochastic iteration step. In stochastic iteration this selection might increase the correlation of the radiance functions of subsequent iteration, but the average will still converge to the expected value according to the Bernstein theorem.

The selection probabilities are found to give higher chances to those methods which are hopefully effective in the current situation. Suppose that in the last iteration step the patches got $I_1(\omega_1), I_2(\omega_2), \ldots, I_n(\omega_n)$ irradiance values (note that directions $\omega_1, \ldots, \omega_n$ are the same for the parallel ray-bundle radiance transfer, but not for perspective ray-bundles and ray shooting).

If all patches have similar powers, we should prefer parallel ray-bundles. The similarity of patch powers can be expressed by the radio of the maximum patch power Φ^{max} and the sum of the powers of all patches total Φ^{total}.

When we decide that non-parallel transfer is applied, the next step is to determine whether not the selected patch is strongly specular. This decision depends on the ratio of the volumes of the diffuse and specular reflection lobes, i.e. of the diffuse albedo $a_{diffuse}$ and the specular albedos a_{spec}, and also on how long the specular reflection lobe is. The length of the lobe can be characterized by the shininess parameter s. Let us define a glossiness function G that maps the used shininess parameters onto $[0, 1]$, in the way that when $G(s)$ is close to 0 then the specular part is highly specular, but when it is close to 1, then it is glossy. An appropriate glossiness function is:

$$G(s) = \frac{1}{1 + \lambda s}$$

where λ should be set to guarantee that a surface of $s = 40$

shininess parameter is considered as the borderline of glossy and highly specular materials, i.e. $G(40) = 0.5$.

In fact we have three options: we consider the surface as diffuse, as glossy or as highly specular. The respective weights of the three options are $a_{diffuse}$, $G(s) \cdot a_{specular}$ and $(1-G(s)) \cdot a_{specular}$. Thus the probability of ray-shooting is:

$$p_{perspective} = \frac{(1-G(s)) \cdot a_{specular}}{a_{diffuse} + a_{specular}}$$

The complete algorithm is:

```
**foreach** patch $i$ **do** $\Phi_i^{out} = I_i(\omega_i) \cdot a_i(\omega_i) \cdot A_i$
$\Phi^{max} = max\{\Phi_j^{out}\}$
$\Phi^{total} = \sum_j \Phi_j^{out}$
$p_{nonparallel} = \Phi^{max} / \Phi^{total}$
Generate a random number $r$ in $[0,1)$
**if** $r < p_{parallel}$
  Select $i$ with probability $p_i = \Phi_i^{out} / \Phi^{total}$
  $p_{perspective} = \frac{(1-G(s)) \cdot a_{specular}}{a_{diffuse} + a_{specular}}$
  Generate a random number $s$ in $[0,1)$
  **if** $r < p_{perspective}$ **then** Use perspective ray-bundle tracing
  **else** Use ray shooting
**else**
  Use parallel ray-bundle tracing
**endif**
```

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

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

The finite-element component $\tilde{L}$ is view independent thus remains valid for the new viewpoint. The emission function L^e should be re-evaluated at each sample point. The only term which poses difficulties is the Monte-Carlo component ΔL.

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. 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.

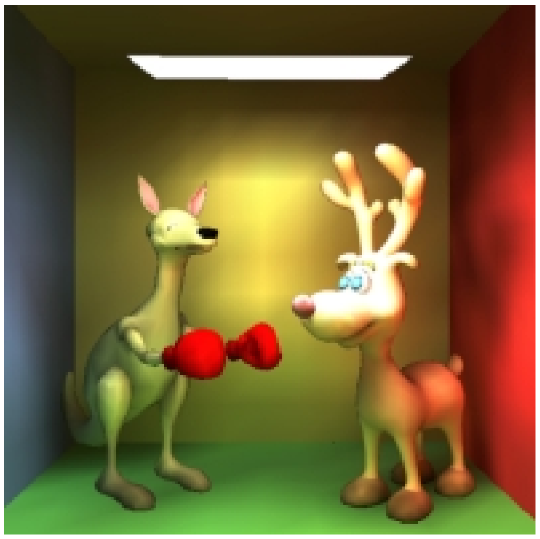

Figure 4: *Animals in the Cornell box*

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.

7. Conclusions

In this paper we proposed a stochastic iteration algorithm that dynamically combines three random radiance transport

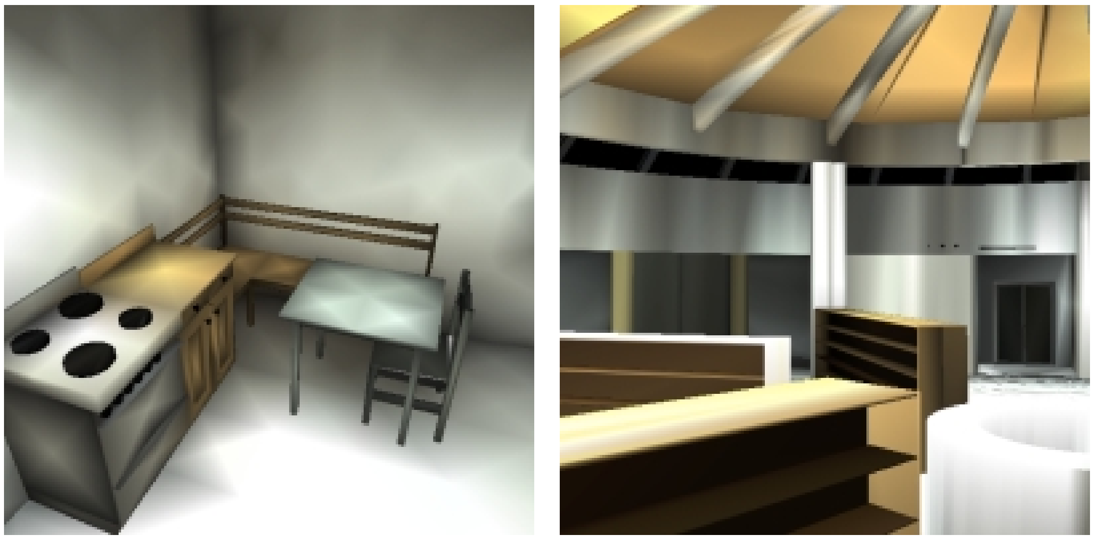

Figure 5: *Snap-shots of a walk-through in a house modelled in ArchiCAD. The images are rendered with the proposed method.*

methods based on the actual radiance approximation. The method is able to render complex glossy scenes in about a minute and is particularly effective if the surfaces are not highly specular. When the image is ready, a rough estimate of the radiance in object space is also available. This estimation requires just one or a few radiance values per patch, thus the storage requirements is modest. Thus when the viewpoint changes, the new image can be generated from this estimation interactively. Note that when the objects themselves move a bit, the radiance representation remains quite accurate, thus this will be a good initial value for the iteration. It means that the proposed method is also good for general animation sequences.

8. Acknowledgements

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T029135), the Eötvös Foundation and the IKTA-00101/2000 project. 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.

References

1. 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.

2. M. Cohen and D. Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Computer Graphics (SIGGRAPH ’85 Proceedings)*, pages 31–40, 1985.

3. 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.

4. 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.

5. 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.

6. J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH ’86 Proceedings)*, pages 143–150, 1986.

7. E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics ’93*, pages 145–153, Alvor, 1993.

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

9. P. Shirley. Time complexity of Monte-Carlo radiosity. In *Eurographics ’91*, pages 459–466. Elsevier Science Publishers, 1991.

10. 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.

11. P. Slussalek. Photo-realistic rendering — recent trends and developments. In *Eurographics '97, STAR reports*, pages 35–57, 1997.

12. 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.

13. 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/script.pdf.

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

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

16. 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.

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