Thirteenth Eurographics Workshop on Rendering (2002)
P. Debevec and S. Gibson (Editors)

Accelerating Path Tracing by Re-Using Paths

Philippe Bekaert
Max-Planck-Institut für Informatik, Saarbrücken, Germany.
Philippe.Bekaert@mpi-sb.mpg.de

Mateu Sbert
Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain.
mateu@ima.udg.es

John Halton
University of North Carolina at Chapel Hill.
halton@cs.unc.edu

Abstract

This paper describes a new acceleration technique for rendering algorithms like path tracing, that use so called gathering random walks. Usually in path tracing, each traced path is used in order to compute a contribution to only a single point on the virtual screen. We propose to combine paths traced through nearby screen points in such a way that each path contributes to multiple screen points in a provably good way. Our approach is unbiased and is not restricted to diffuse light scattering. It complements previous image noise reduction techniques for Monte Carlo ray tracing. We observe speed-ups in the computation of indirect illumination of one order of magnitude.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture; Ray tracing

1. Introduction

The generation of photo-realistic images has been one of the main goals of computer graphics for over 25 years. The algorithms that produce the most convincing images today, are based on a simulation of light transport according to the laws of physics. Such illumination simulation requires to solve the rendering equation [10], a Fredholm integral equation of the second kind. Both deterministic and Monte Carlo methods have been proposed to tackle this problem. To date, Monte Carlo algorithms appear to deal significantly more easily with complex scenes, containing for instance procedurally defined geometry exhibiting non-diffuse light emission and scattering.

One can distinguish basically three kinds of Monte Carlo global illumination algorithms: 1) algorithms in which the trajectory of light particles originating at light sources is simulated according to the laws of physics (shooting algorithms), 2) algorithms simulating the path of similar imaginary particles originating at the observer position rather than the light sources (gathering algorithms), and 3) hybrid algorithms in which paths are traced both from the light sources and from the observer position. The prototype example of a gathering algorithm is stochastic ray or path tracing [5, 10]. Bidirectional path tracing [15, 24] and two-pass algorithms such as photon mapping [9] are examples of a state-of-the-art hybrid algorithms.

This paper focuses on gathering algorithms such as (but not restricted to) path tracing. The computational cost of such algorithms can be very high: usually, hundreds of random walks need to be traced per screen pixel in order to obtain a smooth image, in which statistical noise is not objectionable to the human eye anymore. Each random walk requires numerous visibility tests to be performed by ray shooting, as well as light emission and scattering distribution evaluations and sampling operations. With a simple shading model, such as the Phong model, visibility testing easily takes over 85% of the total computation time. State-of-the-art PC's allow to shoot hundreds of thousands of rays per second. Still, computing a medium resolution image (e.g. 640×480 pixels) can take hours this way.

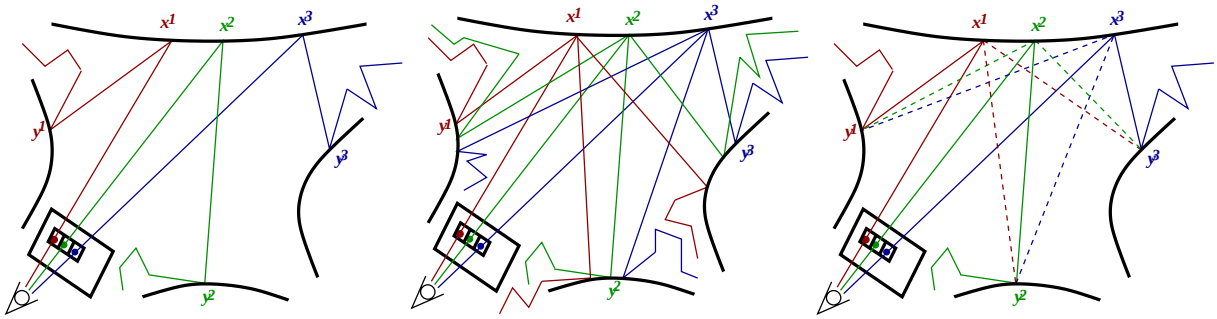

Figure 1: *The basic idea of the paper: (left) in path tracing, the illumination received in each image pixel is computed by tracing a path through each pixel; (center) noise is reduced by tracing multiple independent paths through each pixel; (right) we propose to re-use paths shot through nearby pixels in order to reduce noise. We do so by tracing a single shadow ray per pair of paths in a group, connecting points x_i with y_j, $i, j = 1,2,3, i \neq j$ as indicated by the dashed lines. Stated roughly, this results in additional paths at the cost of single rays. With appropriately chosen combination weights, no bias is introduced.*

The cost of such algorithms can be reduced by either reducing the number of paths to be traced in order to obtain an acceptable image, or by reducing the cost per traced path. Techniques such as adaptive pixel super-sampling [17, 14], path differentials [8, 22], low discrepancy and interleaved sampling [11, 13], hybrid algorithms such as bi-directional path tracing and two-pass methods such as photon mapping, aim at a reduction of the number of paths to be traced. The latter obtain this effect in part by re-using paths traced from light sources for multiple measurements to one or more screen pixels. Metropolis light transport[25] is an extreme example: subsequently traced paths often do not differ in more than one path vertex, which has been changed according to some mutation strategy.

Russian roulette and splitting [1, 3] aim at a reduction of the cost per path by tuning the path depth and branching factor. Interpolation and smoothing techniques such as irradiance caching [27], anisotropic diffusion [16] and the discontinuity buffer [12] reduce the cost per path by amortizing the cost over different screen pixels. Also the adaptive image filtering techniques in [19, 23] can be interpreted in this way.

In this paper, we present a new technique to amortize the cost of path tracing over several screen pixels. Rather than smoothing or interpolating the energy in nearby screen locations, we propose to combine paths traced through nearby screen locations so that each path yields a contribution to multiple screen points (see figure 1). We will show that this can be done for non-diffuse light scattering, and without introducing bias. The principle was proposed before by the third author in the context of sequential Monte Carlo methods for solving of linear systems[7]. Note that unlike hybrid and two-pass algorithms, in which gathering paths are combined with shooting paths, only gathering paths are involved here.

In §2, we present the basic idea in general. Next, we will work out its application to stochastic ray tracing in §3.

2. The principle of re-using gathering random walks

We first discuss the principle of gathering path re-use in the context of the solution of a system of linear equations, such as the radiosity system of equations. In §2.4, we generalize the principle to the solution of integral equations like the rendering equation. The re-use of random walks, as presented here, can be applied wherever gathering random walks are used. It is certainly not restricted to radiosity or ray tracing.

2.1. Gathering Random Walk Estimators

First, we will explain how a system of linear equations can be solved using gathering random walks. Consider a system of linear equations

$$x_i = \sum_{j=1}^{N} H_{ij} x_j + a_i \qquad i = 1, \ldots N. \tag{1}$$

Consider now a sequence of indices $\gamma_r \in 1, \ldots, N$, constructed as follows: the first index γ_1 is chosen from the set $1, \ldots, N$ with *birth probability* R_{γ_1}. The second index γ_2 is chosen conditional on γ_1 with *transition probability* p_{γ_1,γ_2}. Subsequent indices γ_r are selected similarly, conditional w.r.t. γ_{r-1} with probability $p_{\gamma_{r-1},\gamma_r}$. A sequence of indices like this is an example of a *random walk*. The indices correspond to the *states* of the random walk. The birth probabilities R_i need to be normalized ($\sum_i R_i = 1$), but the transition probabilities can be *sub-critical*: in case $\sum_j p_{ij} < 1$ for a certain index i, the construction of the sequence is terminated at i with *absorption probability* $\alpha_i = 1 - \sum_j p_{ij} > 0$. The probability associated with the whole sequence $\bar{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_l]$, is $p(\bar{\gamma}) = R_{\gamma_1} p_{\gamma_1,\gamma_2} \ldots p_{\gamma_{l-1},\gamma_l} \alpha_{\gamma_l}$.

One can also associate a value, or *score*, with a random walk $\bar{\gamma}$. For instance:

$$g_r^G(\bar{\gamma}) = \frac{\delta_{r,\gamma_1} H_{\gamma_1,\gamma_2} \ldots H_{\gamma_{l-1},\gamma_l} a_{\gamma_l}}{p(\bar{\gamma})}.$$

δ_{r,γ_1} denotes Kroneckers delta (1 if $r = \gamma_1$ and 0 otherwise). It can be shown that the expected value $E[g_r^G] = \sum_{\overline{\gamma}} g_r^G(\overline{\gamma})\, p(\overline{\gamma})$ of the scores above, for arbitrary random walks $\overline{\gamma}$ constructed as outlined, equals x_r if certain conditions [6, 18] are fulfilled. One important condition is that p_{ij} shall not be zero if H_{ij} isn't. In short: by generating random walks $\overline{\gamma}$ and averaging their scores $g_r^G(\overline{\gamma})$, one can solve the system of linear equations (1).

The procedure outlined here is called gathering, and g_r^G is called a *gathering estimator*, because it uses random walks starting from a state r where one wants to "measure" the solution of the system. The scores shown above are non-zero only when the source term a_{γ_l} of the system is non-zero at the state γ_l where the walk is terminated. Because it estimates only at absorption, it is called an *absorption estimator*. g_r^G is also called an *unbiased* estimator, because its expected value equals x_r exactly. It is but one example of a random walk estimator for linear systems. Many more such estimators are described in literature [6, 18].

2.2. Re-using gathering random walks

Let us now define a new estimator g_i for x_i in (1) in the following way. We select a term γ_1 in the sum in (1) with probability R_{γ_1}, and a path $\overline{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_r, \ldots]$ with probability $p(\overline{\gamma})$, and define:

$$g_i = \frac{H_{i\gamma_1}}{R_{\gamma 1}} g_{\gamma_1}^G + a_i \tag{2}$$

where $g_{\gamma_1}^G$ is an arbitrary gathering random walk estimator for x_{γ_1} using path $\overline{\gamma}$. The new estimator g_i will be unbiased if *i)* $R_j > 0$ whenever $H_{ij} \neq 0$ and *ii)* g^G is unbiased.

Proof: (sketch of —) The proof follows directly from the premises and the definition and general properties of expected values: $E[g_i] = \sum_{\gamma 1} [(H_{i\gamma_1}/R_{\gamma 1}) E[g_{\gamma_1}^G] + a_i] R_{\gamma 1} = \sum_{\gamma 1} H_{i\gamma_1} x_{\gamma 1} + a_i = x_i$.

The benefit of the new estimator can be understood as follows: consider another index k, so that $R_j > 0$ whenever $H_{kj} \neq 0$. Then, g_k can be sampled simultaneously with g_i. In general, if S_R is the set of states that fulfill this condition, then all states in S_R can be estimated simultaneously. This is the same as saying that one path $\overline{\gamma}$ can be reused for all states in S_R, and one state in S_R can reuse all paths generated from states γ_1 with $R_{\gamma_1} > 0$.

Particular cases:

If we take $R_j > 0$ for all $j = 1 \ldots N$, then S_R covers all indexes (states). An example of this is the uniform distribution, $R_j = \frac{1}{N}$ where N is the number of indexes (states).

Another possibility is to fix a particular state h and to take $R_j = p_{hj}$. In this case, the set S_R contains the states k that fulfill the following condition: all states j, for which $H_{kj} \neq 0$, can be reached by sampling a random walk transition from h: $p_{hj} > 0$.

2.3. Application to Radiosity

In principle, it is possible to apply the new estimator to the radiosity problem. In the radiosity method [4, 21], the following system of linear equations needs to be solved:

$$B_i = \sum_j \rho_i F_{ij} B_j + E_i. \tag{3}$$

B_i denotes the radiosity of a patch i, E_i the emissivity, ρ_i the reflectivity and F_{ij} the (unknown) form factor between patches i and j.

During the last 12 years, numerous Monte Carlo algorithms have been proposed for the radiosity problem. Many of these solve the linear system above by means of random walks [2]. The use of gathering random walks in this context is discussed in [20]. Monte Carlo radiosity algorithms are all based on the observation that the form factors can be interpreted as probabilities, which, although unknown and costly to compute, are easy to sample and to use as transition probabilities for sampling random walks. In this way, one can avoid having to compute the numerical value of form factors and to store form factors or links between patches.

Estimator (2) becomes in this context

$$g_i = \frac{\rho_i F_{i\gamma_1}}{R_{\gamma 1}} g_{\gamma_1}^G + E_i \tag{4}$$

Using this estimator in a naive way however, implies computing form factors explicitly, which is what one wants to avoid by using a random walk algorithm.

2.4. Integral equations

Fredholm integral equations of the second kind

$$f(X) = \int_D H(X,Y)\, f(Y)\, dY + a(X). \tag{5}$$

can be solved in a similar way. The states of the random walks now correspond with points X and Y in the domain D of the integral, rather than with indices i or j before. The discrete probabilities of before now become continuous probability densities. The estimator corresponding to (2) is

$$g(X) = \frac{H(X,\gamma_1)}{R(\gamma_1)} g^G(\gamma_1) + a(X). \tag{6}$$

Again, $g^G(\gamma_1)$ can be any random walk gathering estimator for $f(\gamma_1)$ using paths $\overline{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_r, \ldots]$. As before, g is unbiased if $R(Y) > 0$ whenever $H(X,Y) \neq 0$ and g^G is unbiased. Similar to the latter particular case in section 2.2, one can fix a point $X^\star$ and take $R(Y) = p(X^\star, Y)$ ($p(X,Y)$ is the transition pdf used to generate paths $\overline{\gamma}$). In order to ensure unbiasedness of first g^G, and then g, the pdf p must fulfill the condition $p(X,Y) > 0$ whenever $H(X,Y) \neq 0$.

3. Application to Path Tracing

Path tracing [10] corresponds to the solution of the rendering equation, an integral equation similar to (5), by means of

gathering random walks. In this section, we work out the application of path re-use, explained above, to path tracing.

3.1. Path tracing

In order to obtain a one-to-one correspondence with equation (5) and (6), we consider the following somewhat unusual form of the rendering equation, relating the radiance $L^{\leftarrow}(x,\omega_x)$ at a scene surface location x coming in from the direction ω_x with the incident radiance $L^{\leftarrow}(y,\omega_y)$ at a different location and direction (y,ω_y):

$$L^{\leftarrow}(x,\omega_x) = L_e^{\leftarrow}(x,\omega_x) + \int_S\int_\Omega H(x,\omega_x;y,\omega_y)L^{\leftarrow}(y,\omega_y)dyd\omega_y \quad (7)$$

$$H(x,\omega_x;y,\omega_y) = \delta(y,h(x,\omega_x))\,f_r(y,-\omega_x,\omega_y)|\omega_y\cdot N_y|. \quad (8)$$

The source term $L_e^{\leftarrow}(x,\omega_x) = L_e^{\rightarrow}(h(x,\omega_x),-\omega_x)$ is the self-emitted radiance towards x emitted from the first point $h(x,\omega_x)$ seen from x in the direction ω_x. The right hand side of the equation contains a double integral: over pairs of surface points and directions in the scene. $\delta(y,h(x,\omega_x))$ is a Dirac delta function, which is zero for all points y that differ from $h(x,\omega_x)$, and which has the property that $\int_S\delta(y,z)F(y)dy = F(z)$ where F is some function of surface points. N_y denotes the surface normal at y. $f_r(y,-\omega_x,\omega_y)$ is the bsdf at y. In the sequel, we will omit the "$\leftarrow$" and let $L(x,\omega_x)$ denote incident radiance.

Image synthesis requires that solid angle integrals like the following are solved for every screen pixel i:

$$L^i = \int_\Omega M^i(o,\omega_o)L(o,\omega_o)d\omega_o. \quad (9)$$

o denotes the observer position, and M^i is a "measurement" or "detector response" function associated with the pixel i. In the easiest case, the measurement function is a box function corresponding to the pixel i. With proper normalization, L^i then is the average radiance towards the observer position o coming through the pixel. A pinhole camera model is assumed. Finite aperture camera models (depth of field), give rise to a double integral: also over for instance the camera film surface.

In path tracing, these equations are solved by simulating random walks originating at the observer position (see figure 2). The states of the random walks are pairs (x,ω_x) of a surface point x and a direction ω_x from where illumination is received at x.

More precisely, the random walks are traced with birth probabilities $\beta^i(o,\omega_o)$ which usually are equal to the pixel measurement function $M^i(o,\omega_o)$ (for instance: uniform pixel sampling). The transition probabilities $p(x,\omega_x;y,\omega_y)$ do not depend on the pixel i. A common choice is:

$$p(x,\omega_x;y,\omega_y) = \delta(y,h(x,\omega_x))\,p_r(y,-\omega_x,\omega_y). \quad (10)$$

They are the product of two factors:

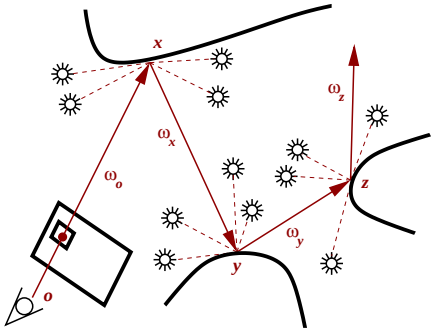

Figure 2: *The random walks in path tracing are sequences of position/direction pairs (o,ω_o), (x,ω_x), (y,ω_y), (z,ω_z), ... as shown in this figure. o corresponds to the viewing position. x is the surface point visible from o along direction ω_o, etc. Light is transported in the reverse directions. In addition, shadow rays (dashed lines) are traced from each path vertex to light source locations (next event estimation).*

1. the factor $\delta(y,h(x,\omega_x))$ indicates that the point y is uniquely determined by the pair (x,ω_x). (Visibility, surface orientation and distance are accounted for implicitly, by the directional parametrization);
2. a sub-critical probability density function $p_r(y,-\omega_x,\omega_y)$ for 1) sampling whether a random walk arriving at y from direction $-\omega_x$ shall be terminated or continued, 2) sampling a scattered direction ω_y if survival is sampled. Ideally, it is equal to the product of the bsdf $f_r(y,-\omega_x,\omega_y)$ times the cosine $|\omega_y\cdot N_y|$.

Since the probability that such a path lands on a light source is small, one traces from each surface point hit by the path, so called shadow rays to the light sources in order to estimate the direct illumination at the hit point (see figure 2). This is called next event estimation in Monte Carlo literature.

3.2. Re-using paths in path tracing

Equation (6) now immediately suggests that incident radiance estimates $\tilde{L}^{\bar{\gamma}}(y,\omega_y)\approx L(y,\omega_y)$ at a location y, using a random walk $\bar{\gamma}$ visiting (y,ω_y), can be re-used for simultaneously estimating the radiance incident from y at a set of different locations x^i:

$$\frac{H(x^i,\omega_{x^iy};y,\omega_y)}{R(y,\omega_y)}\tilde{L}^{\bar{\gamma}}(y,\omega_y) + L^e(x^i,\omega_{x^iy}) \approx L(x^i,\omega_{x^iy}).$$

Note that the direction of incidence ω_{x^iy} at x^i is fixed: it needs to be the direction from x^i to y. As direct illumination is estimated by explicit light sampling, we will omit the source term L^e in the sequel here.

We will apply this idea in the following way: consider, to start with, a contiguous set of $n_v\times n_h$ pixels, which we call a $n_v\times n_h$ image tile τ. Through each pixel k in the tile, we

will trace a path $(o, \omega_o^k), (x^k, \omega_x^k), (y^k, \omega_y^k), \ldots$, exactly in the same way as in standard path tracing. o denotes the observer position, x^k is the surface point directly visible in pixel k, along the direction ω_o^k, ω_x^k is a scattered ray direction at x^k, y^k is the nearest surface location hit by this scattered ray, and ω_y^k is the scattered ray direction at y^k. The theory thus prescribes that it might be possible to re-use a random walk traced through a first pixel i in order to compute the pixel intensity in each other pixel j in the tile. Vice versa, the pixel intensity in each pixel i might possibly be computed using all paths traced through each of the pixels j in the tile:

$$\tilde{L}(x^i, \omega_{x^i y^j}) = \frac{H(x^i, \omega_{x^i y^j}; y^j, \omega_y^j)}{R^j(y^j, \omega_y^j)} \tilde{L}^{\bar{\gamma}}(y^j, \omega_y^j).$$

In order to estimate the indirect radiance emitted from x^i towards the observer position o (needed in (9)), the above estimates for incident radiance to x^i from each y^j are combined as follows:

$$\tilde{L}^{ind}(o, \omega_o^i) = \sum_{j \in \tau} w_{ij} \cdot f_r(x^i, -\omega_o^i, \omega_{x^i y^j}) |\omega_{x^i y^j} \cdot N_{x^i}| \times \frac{H(x^i, \omega_{x^i y^j}; y^j, \omega_y^j)}{R^j(y^j, \omega_y^j)} \tilde{L}^{\bar{\gamma}}(y^j, \omega_y^j). \quad (11)$$

Unbiased estimation follows if i) the weights w_{ij} sum up to 1, ii) $\tilde{L}^{\bar{\gamma}}(y, \omega_y)$ is an unbiased estimator for the incident radiance $L(y, \omega_y)$, and iii) the probabilities $R^j(y, \omega_y)$ are non-zero whenever $H(x^i, \omega_{x^i y}; y, \omega_y)$ is non-zero.

3.3. The probabilities $R^j(y, \omega_y)$ and weights w_{ij}

The probabilities $R^j(y, \omega_y)$ are dictated by the application of path tracing from the points x^j. They are $R^j(y, \omega_y) = p_r(x^j, -\omega_o^j, \omega_{x^j y}) p(x^j, \omega_{x^j y}; y, \omega_y)$. Note that it does not matter how the points x^j and directions ω_o^j are obtained. They are obtained by pixel sampling in our case, but could also be obtained in different ways in other applications. Note also that these expressions give rise to ratios of Dirac pulse functions. They are an artifact of the parametrization and shall be taken equal to 1.

Unfortunately, these probabilities do not in general allow unbiased estimation of $L(x^i, \omega_{x^i y})$ at pixels $i \neq j$, since they may vanish when $H(x^i, \omega_{x^i y}; y, \omega_y) \neq 0$. This can happen because of occlusion (point x^j does not "see" all points y visible from x^i), due to surface orientation (some surfaces facing x^i are facing away from x^j), or due to the bsdf/pdf's at the involved surfaces. Taking equal weights $w_{ij} = 1/(n_v \times n_h)$, or using classical techniques for combining Monte Carlo estimators with fixed weights w_{ij} based on the co-variance matrix [18], will therefore not work.

Fortunately, it is possible to obtain unbiased estimation using averages (11) with non-constant weights $w_{ij}(y, \omega_y)$, as long as for every (y, ω_y) that can contribute illumination to x^i, there is at least one j with non-zero associated $R^j(y, \omega_y)$.

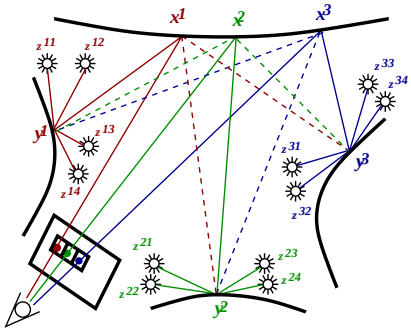

Figure 3: *Shadow rays at the secondary hit points y^j of the paths can be re-used easily as well for computing first order indirect illumination at the points x^i. If 4 shadow rays are traced from every y^j (as shown in this illustration), each connection between two paths (dashed lines) yields 4 new samples for first order indirect illumination at the x^i. Note that the required visibility tests are already done for re-using the scattered path at the y^j points (see figure 1-right).*

This is, of course, always the case: by tracing paths from x^i itself, no non-zero illumination contributions to x^i are overlooked. Note that combination with (11) reduces to standard path tracing when the weights are chosen $w_{ij} = \delta_{ij}$. Veach [24] has proposed provably good and practical heuristics for chosing weights in such situations. His balance heuristic, for instance, in our case translates to

$$w_{ij}(y, \omega_y) = \frac{R^j(y, \omega_y)}{\sum_{k \in \tau} R^k(y, \omega_y)}. \quad (12)$$

Besides a visibility test, each connection thus requires to evaluate the distance, cosines, and the bsdf and scattering pdf at both connected path vertices x^i and y^j. The weights (12) are evaluated for (y^j, ω_y^j). For fixed j, they turn out to be equal for all i.

In the appendix, we outline a more technical derivation of this result, from an alternative point of view.

3.4. Explicit light sampling

As explained before, the efficiency of path tracing can be very low unless explicit light sampling (next event estimation) is carried out at each path vertex, including all y^j. The light samples z^{jl} at y^j can be used for the perturbed directions to all x^i in the tile, in addition to the direction to x^j, where the path came (see figure 3). No additional visibility tests are required. It is sufficient to re-evaluate the bsdf f_r at y^j for each pair of light source sample point z^{jl} and ancestor point x^i. The combination weights are identical to the weights (12), except that they now contain the probabilities $p_l(y^j; z^{jl})$ by which light source points z^{jl} are sampled, rather than the probabilities $p_r(y^j, ., .)$ of BSDF-based sampling at y^j (see appendix).

3.5. Choosing an appropriate tile size

The choice of the tile size is subject to two contradicting requirements: on the one hand side, one will obtain a more prominent variance reduction by chosing a large tile size, thus re-using many paths. On the other hand, a smaller tile size yields a lower additional cost as fewer BSDF evaluations and visibility tests need to be done per path, and leads to more effective combinations due to higher coherence at different levels:

- First, the smaller the distance between the points x^i, the higher the chance that the pairs of points (x^i, y^j) will be unoccluded;
- Second, shadow rays to be traced for performing the visibility tests originating at each point y are aimed into highly similar directions and will take less computation time since parts of the scene to be traversed will already be present in high speed memory caches. These shadow rays could for instance be traced efficiently in bundles [26];
- Third, the directions $\omega_{x^i y^j}$ will be close to the directions $\omega_{x^i y^j}$ obtained by BSDF-based sampling;
- Fourth, the correlation between the radiance estimates at different pixels in a tile manifests itself in the form of reduced noise compared to independent estimation (positive co-variance).

The next section describes a heuristic that can help to choose an appropriate tile size.

3.6. Efficiency improvement

The efficiency gain that can be obtained by re-using paths depends on the average path length l, the number k of explicit light samples at each path vertex and the number N of paths that are combined in a group. Let's measure the cost as the number of visibility tests to be performed. The cost of tracing a single path is then on the average $l + lk$. For N paths, the cost is $Nl(k+1)$. Re-using the paths in groups of N takes an additional $N(N-1)$ visibility tests, leading to a total cost of $Nl(k+1)+N(N-1)$. Now suppose that path re-use would yield the same accuracy as tracing Nm independent paths, with $1 \leq m \leq N$. The cost of tracing Nm independent paths is $Nml(k+1)$. The gain can then be estimated as

$$\nu = \frac{l(k+1)\cdot m}{l(k+1)+(N-1)}.$$

Example: suppose the average path length $l \approx 4$. With $N = 16, k = 16$ and a number of effective combinations $m = 12$, the gain would be $\nu \approx 9.8$. The ratio m/N of course depends on the scene, view and image resolution.

In addition, there is a positive covariance of the estimators, which exhibits itself in a reduced difference between the pixels pierced by the connected paths.

On the other hand, the gain is not equally large everywhere. In particular, re-using paths when $H(x,\omega_x;y,\omega_y)$ is small, yields little benefit. This is the case when the bsdf is small, or when surfaces are viewed at grazing angles, for instance near certain edges. The efficiency can be improved in such situations by reducing the number of visibility tests. This could be done by using Russian roulette like techniques [1] based on the combination weights (12).

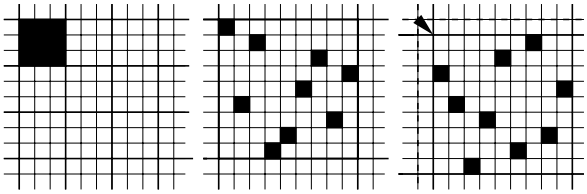

Figure 4: *In order to make noisy artifacts in the images less noticeable, we recommend to combine pixels in N-rooks like patterns (middle) rather than in solid rectangles (left). When the tiles are also shifted in subsequent passes through the image (right), their boundaries quickly become invisible.*

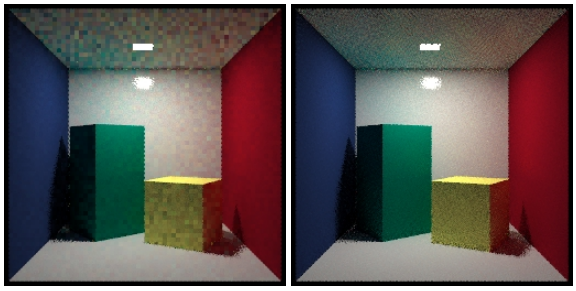

Figure 5: *These images have both been obtained with 2 paths per pixel, combining paths in groups of 16. The only difference is in the arrangement of the paths that are combined: 4×4 pixel blocks (left) and in N-rooks configurations (right). The images have similar average numerical error (the left one should even have slightly lower error), but perceptually, the right image is clearly better.*

3.7. Making noise less visible

A fixed subdivision of the screen in rectangular tiles, results in disturbing discontinuities at the tiles edges. These discontinuities only disappear for a large number of paths per pixel. It seems that the noise patterns that result from naive application of our method are of a frequency for which the human eye is more sensitive. Although numerically, the error may have decreased significantly, perceptually, the error remains large.

We implemented two simple measures in order to translate the resulting low-frequency noise to less noticeable noise patterns of a higher frequency (see figure 4 and 5):

- we combine pixels in N-rooks like configurations rather than in solid rectangles;

- the image tiles are shifted in subsequent passes through the image, so that their boundaries are in different places. For instance, by shifting 8x8 tiles 1 pixel horizontally and vertically in each pass, the boundaries totally disappear after 8 passes.

These measures hardly affect the numerical error of the resulting images. Instead, they shift noisy patterns to a frequency band for which the human visual system is less sensitive (see figure 5).

4. Results

Figures 6 and 7 show some results obtained with the algorithm described above. The simple scene in figure 6 allows to compute a highly accurate solution within reasonable time (one night). This reference solution has been used for detailed studies of the error reduction rate obtained with the algorithm. The figure clearly shows the improvement in diffuse and glossy indirect illumination when re-using paths. The reduction of the mean square difference w.r.t. the converged solution was about a factor of 9, largely independent of the sampling rate, but of course depending on the number of paths that is combined (16 in this case). Image 6f shows that the error reduction is, as predicted, not uniform: near object edges, re-using paths can become slightly less effective because paths shot through neighboring pixels do not provide useful information about the illumination above the surface hit by the first path. Still, there is a significant improvement even near edges, as can be judged from figures 6c and 6d. The additional cost of path re-using in this simple scene is however quite large: about 83% when combining paths in groups of 16, due to the number of additional BSDF and PDF evaluations (32 per pixel). Timings were 54 seconds without path re-use and 99 seconds with path re-use for the shown 256×256 pixel images 6c and 6d, on a 850MHz Pentium-III based laptop computer. The shown example is a worst-case example: both ray shooting and light source sampling (only a single light sample per path vertex; only one light source) are extremely cheap in this case. Even in this worst case however, there still is a considerable nett gain: factor 9 variance reduction, divided by 99/54 = 4.9 nett speed-up.

Figure 7 shows a more complex example: a building floor model with 315,000 polygons of which 2450 are light sources. The high number of light sources necessitates importance sampling of direct illumination at path vertices. At every path vertex, such importance sampling requires 10 floating point additions and 10 multiplications per light source in order to calculate the light selection probabilities. This cost was amortized over 16 explicit light samples at each path vertex. Because of the high cost of explicit light sampling, the overhead of re-using paths is much less significant than in the previous, simple, example: only about 15%. The overhead is mainly due to the additional BSDF evaluations for re-using the light samples. The computation time

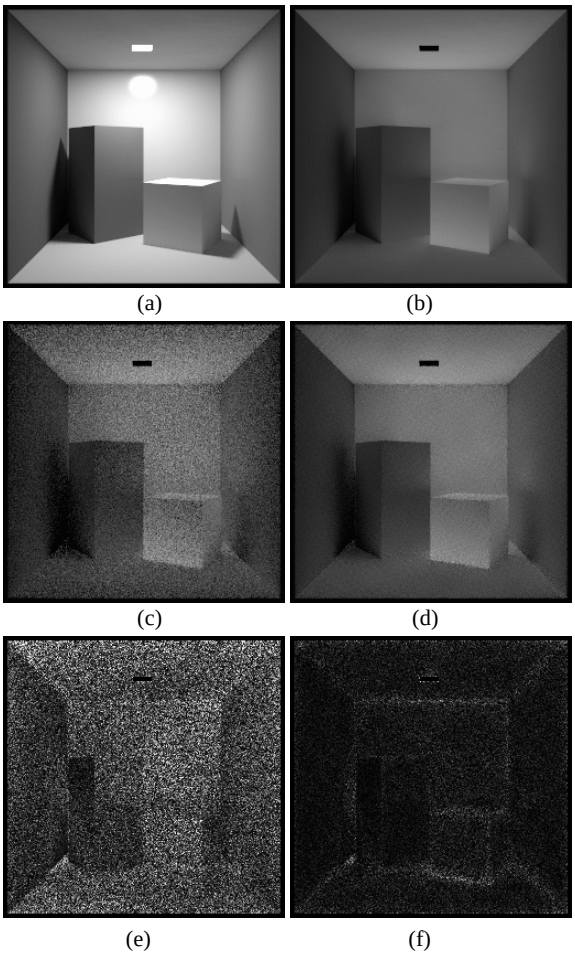

Figure 6: *Results obtained by re-using paths: (a) converged image showing all direct illumination as well as glossy and diffuse indirect illumination; (b) converged image showing only glossy and diffuse indirect illumination; (c) image obtained with 15 paths per pixel without re-using paths; (d) image obtained with 15 paths per pixel with re-use of paths in groups of 16, configured as 16 rooks on 16×16 tile grids; (e) difference of (c) with (b), 5 times exaggerated; (f) difference of (d) with (b), 5 times exaggerated. Re-using paths in this example, and as indicated here, results in a reduction of the mean square error by a factor of 9. The reduction of noisy artefacts is clearly visible in the images.*

for the shown images (512×384 pixels, 100 paths per pixel) was about 8 hours. The desks and desk lamps have been assigned non-diffuse reflection properties.

The images show that re-using paths remains highly effective at reducing noisy artefacts in complex scenes too. It was not possible to measure the speed-up in this scene, like for the simple cube scene above, because obtaining a sufficiently converged reference image for such measurements

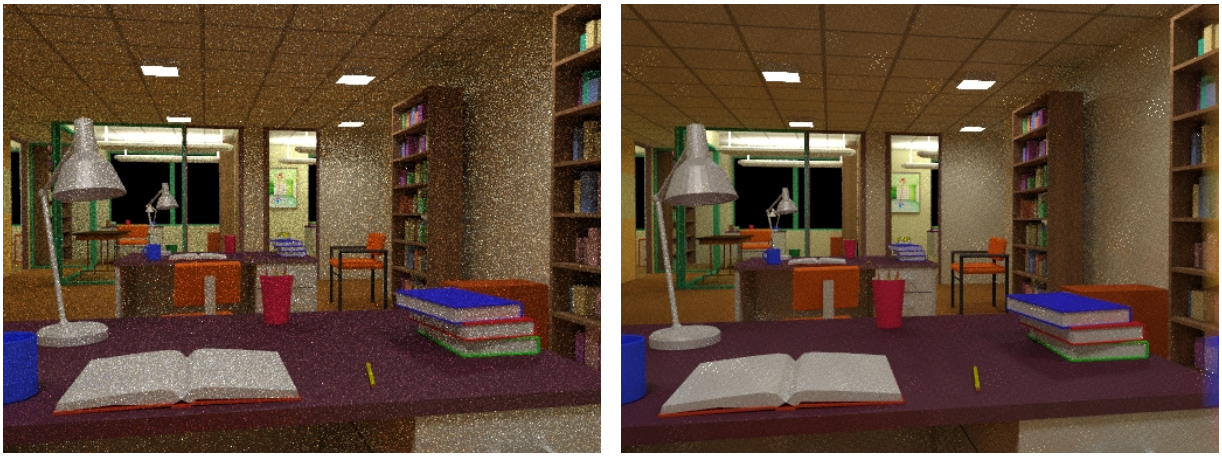

Figure 7: *Left: image without re-using paths; Right: image with path re-use. 100 paths per pixel were traced, with 16 explicit light samples at each path vertex. The paths were combined in groups of 16 in the right image. The images show a view in a building floor model consisting of about 315,000 polygons of which 2450 are light sources. The high number of light sources necessitates importance sampling strategies for light source selection at every path node. The CPU time overhead of path re-use in this example is about 15%. The desks and desk lamps are non-diffuse reflecting surfaces. These images show that re-using paths remains very effective at reducing noisy artefacts in complex environments.*

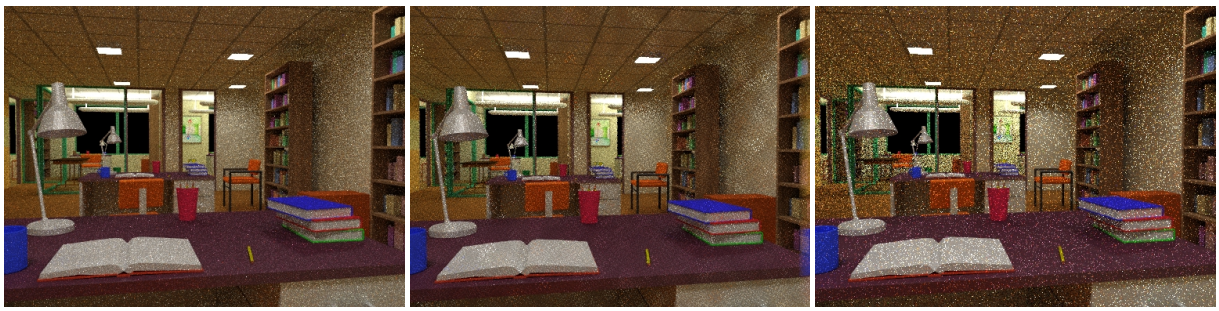

Figure 8: *Left: image obtained with 160 paths per pixel, not re-using paths. Middle: 12 paths per pixel, with path re-use. The parameters are the same as in figure 7. The computation time was about 12 times lower than for the left image. Right: 12 paths per pixel, not re-using paths. We estimate that the speed-up caused by re-using paths is about one order of magnitude.*

would take too much time. By making visual comparisons of images obtained for different number of samples, we estimate the improvement is of about one order of magnitude (see image 8).

Figure 7-Right however still exhibits some disturbing spike noise. Indeed: path re-using does not remedy the fact that stochastic ray tracing tends to under-sample certain light transport paths involving multiple non-diffuse scattering events. Path re-use even tends to make such spike noise more visible to the human eye, as the noise is distributed over several pixels in an easily spotted pattern. Such spike noise is highly reduced in more advanced, hybrid, algorithms like bi-directional path tracing. Also, image smoothing in addition to our technique, will be highly effective at eliminating such spike noise.

5. Conclusion

In this paper, we brought to the attention a variance reduction technique for Monte Carlo solution of linear systems and integral equations, that was previously proposed by the third author [7]. We demonstrated its application to stochastic ray tracing: by using traced paths, not only for computing an image contribution at the pixel through which the path was shot, but also for neighboring pixels, the cost of path tracing is amortized over several pixels. We showed that this can be done for non-diffuse light scattering, and without introducing bias. We observed speed-ups of about one order of magnitude.

The principle of path re-use as explained in this paper is not restricted to path tracing. It can be integrated in hybrid algorithms too, like bi-directional path tracing or photon

mapping. In such algorithms, it will allow to re-use gathering random walks in addition to shooting random walks. In the context of photon mapping, the proposed technique may speed up final gathering by allowing expensive nearest neighbor photon queries to be re-used. In bi-directional path tracing, bi-directional paths through different pixels can be combined.

The main problem of the proposed technique is that it tends to make spike noise in the images more visible, since such noise is distributed over several neighboring pixels in an easily spotted pattern. This noise can however be removed by applying previously proposed image filtering techniques for Monte Carlo path tracing in addition to our new technique. We expect that smoothing an image obtained with path re-use will work just as well as it does for images computed using plain Monte Carlo ray tracing because the noise patterns still are sufficiently random. We also expect that our technique will combine very well with interleaved sampling [13].

Acknowledgments

This project started during a visit of John Halton to the University of Girona in June 2001. Mateu Sbert is partially supported by Grants TIC-2001-2416-C03-01 of the Spanish Government, 2001-SGR-00296 of the Catalan Government and the Spanish-Austrian Joint Action number HU2000-0011. Philippe Bekaert acknowledges financial support by a Marie Curie post-doctoral research fellowship from the European Commission (contract nr. HPMF-CT-2001-01361). The model used in figures 7 and 8 is the SODA hall model from the university of Berkeley. The authors are grateful to Alexander Keller for commenting on an earlier version of this text. Ingo Wald, Philipp Slusallek and Carsten Benthin were so kind to provide access to their OpenRT ray tracing engine for our implementation. An early implementation of the method was performed in the context of the RenderPark system for global illumination, available from `www.renderpark.be`.

References

1. J. Arvo and D. Kirk. Particle transport and image synthesis. In *SIGGRAPH'90 Conference Proceedings*, pages 63–66, August 1990. 2, 6
2. Ph. Bekaert. *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, K. U. Leuven, Leuven, Belgium, December 1999. 3
3. M. A. Bolin and G. W. Meyer. An error metric for Monte Carlo ray tracing. In *Eurographics Workshop on Rendering '97*, pages 57–68, June 1997. 2
4. M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993. 3
5. R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics (SIGGRAPH' 84 Proceedings)*, 18(3):137–145, July 1984. 1
6. J. H. Halton. A restrospective and prospective survey of the Monte Carlo method. *SIAM Review*, 12(1):1 – 63, January 1970. 3
7. J. H. Halton. Sequential Monte Carlo techniques for the solution of linear systems. *Journal of Scientific Computing*, 9(2):213–257, 1994. 2, 8
8. H. Igehy. Tracing ray differentials. In *Proceedings of SIGGRAPH 99*, pages 179–186, 1999. 2
9. H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, 2001. 1
10. J. T. Kajiya. The rendering equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):143–150, August 1986. 1, 3
11. A. Keller. Quasi-Monte Carlo radiosity. In *Eurographics Rendering Workshop 1996*, pages 101–110, 1996. 2
12. A. Keller. *Quasi-Monte Carlo methods for image synthesis*. PhD thesis, University of Kaiserslautern, Germany, June 1997. 2
13. A. Keller and W. Heidrich. Interleaved sampling. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 269–276, 2001. 2, 9
14. D. Kirk and J. Arvo. Unbiased sampling techniques for image synthesis. In *SIGGRAPH '91 Conference Proceedings*, pages 153–156, July 1991. 2
15. E. P. Lafortune and Y. D. Willems. Bi-directional path tracing. In H. P. Santo, editor, *Compugraphics '93 Conference Proceedings*, pages 145–153, Alvor, Portugal, December 1993. 1
16. M. D. McCool. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Transactions on Graphics*, 18(2):171–194, April 1999. 2
17. W. Purgathofer. A statistical method for adaptive stochastic sampling. *Comput. Graph.*, pages 157–162, August 1987. 2
18. R. Y. Rubinstein. *Simulation and the Monte Carlo method*. J. Wiley and sons, 1981. 3, 5
19. H. E. Rushmeier and G. J. Ward. Energy preserving non-linear filters. In *Proceedings of SIGGRAPH 94*, pages 131–138, 1994. 2
20. M. Sbert. Error and complexity of random walk Monte Carlo radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):23–38, March 1997. 3
21. F. Sillion and C. Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994. 3

22. F. Suykens and Y. Willems. Path differentials and applications. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 257–268, 2001. 2

23. F. Suykens and Y. D. Willems. Adaptive filtering for progressive Monte Carlo image rendering. In *Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000)*, Plzen, Czech Republic, February 2000. 2

24. E. Veach and L. J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *SIGGRAPH 95 Conference Proceedings*, pages 419–428, August 1995. 1, 5, 10

25. E. Veach and L. J. Guibas. Metropolis light transport. In *SIGGRAPH 97 Conference Proceedings*, pages 65–76, August 1997. 2

26. I. Wald, P. Slusallek, C. Benthin, and M. Wagner. Interactive rendering with coherent ray tracing. *Computer Graphics Forum*, 20(3):153–164, 2001. 6

27. G. J. Ward and P. Heckbert. Irradiance gradients. In *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992. 2

Appendix: alternative derivation

In this appendix, we sketch an alternative derivation of the technique presented in this paper. We start from the following, more familiar, form of the rendering equation:

$$L(x \to o) = L^e(x \to o) + \int_S f_r(x; o \leftrightarrow y) G(x,y) L(y \to x) dy.$$

$L(x \to o)$ denotes the radiance emitted at x in the direction to o. L^e is the self-emitted radiance. The integral is over the surfaces in the scene. We use a spatial parametrization to make all geometric factors (implicit in the main text) visible. $f_r(x; o \leftrightarrow y)$ is the BSDF at x for directions from x to o and y. $G(x,y) = |\omega_{xy} \cdot N_x| g(x,y)$ with $g(x,y) = |-\omega_{xy} \cdot N_y| \mathrm{vis}(x,y) / r_{xy}^2$ is the usual differential throughput factor between x and y ($\mathrm{vis}(x,y)$ is the visibility predicate and r_{xy}^2 is the square distance).

The indirect radiance observed in a pixel i can now be written as a triple integral over the surfaces S:

$$L_i^{ind} = \int_S \int_S \int_S M^i(x) G(o,x) f_r(x; o \leftrightarrow y) G(x,y) f_r(y; x \leftrightarrow z) G(y,z) L(z \to y) dz dy dx$$

o denotes the observer position. $M^i(x)$ is a pixel measurement function for pixel i, expressed in terms of surface points rather than directions. For ease, we assume the observer corresponds to an infinitesimal surface with normal equal to the viewing direction, so we can use the same throughput factor $G(o,x)$ for the observer position as for all other points (with appropriate changes, the derivation remains valid if this would not be the case).

By separating $L(z \to y)$ in two terms: self-emitted radiance $L^e(z \to y)$ and non-selfemitted radiance $L^{ne}(z \to y)$, L_i^{ind} can be expressed as the sum of two integrals L_i^1 and L_i^2. The first integral, L_i^1 contains $L^e(z \to y)$ instead of $L(z \to y)$ and corresponds with first order indirect pixel radiance. The second integral L_i^2 contains the non-selfemitted radiance $L^{ne}(z \to y)$ and corresponds to higher order indirect pixel radiance. These integrals are sampled in different ways. For the first one, points z are obtained by light source sampling, with pdf $p_l(y;z)$, which in the most common case does not depend on y, and is just proportional with the self-emitted radiosity at z. For the second one, points z are obtained by BSDF sampling at y, and scattered rays are sampled at z and subsequently visited points in order to obtain an estimate for $L^{ne}(z \to y)$.

The path re-use weights for first order indirect pixel radiance (section 3.4) can now be derived as follows. Consider

$$L_i^1 = \int_S \int_S \int_S M^i(x) G(o,x) f_r(x; o \leftrightarrow y) G(x,y) f_r(y; x \leftrightarrow z) G(y,z) L^e(z \to y) dz dy dx.$$

In path tracing, this integral is estimated as

$$L_i^1 \approx \frac{M^i(x)}{\beta^i(x)} \frac{f_r(x; o \leftrightarrow y) G(x,y)}{p_r(o,x;y) g(x,y)} \frac{f_r(y; x \leftrightarrow z) G(y,z) L^e(z \to y)}{p_l(y;z)}. \quad (13)$$

$\beta^i(x)$ (pixel sampling) and $p_r(o,x;y)$ (sampling a direction for scattering) are the same pdf's as in the main text, but expressed in terms of surface locations instead of directions. Because the pixel measurement function integrates to 1 for all pixels, the triple integral above is also equivalent to the following quadruple integral:

$$L_i^1 = \int_S \int_S \int_S \int_S M^j(x') M^i(x) G(o,x) f_r(x; o \leftrightarrow y) G(x,y) f_r(y; x \leftrightarrow z) G(y,z) L^e(z \to y) dz dy dx dx'.$$

$M^j(x')$ can be the measurement function for a different pixel j. The point x' can be sampled using pixel-j sampling (pdf $\beta^j(x')$, usually identical to $M^j(x')$). y can be sampled by tracing a random walk from x' rather than x. Doing so yields to following estimates:

$$L_i^1 \approx \frac{M^j(x')}{\beta^j(x')} \frac{M^i(x)}{\beta^i(x)} \frac{f_r(x; o \leftrightarrow y) G(x,y)}{p_r(o,x';y) g(x',y)} \frac{f_r(y; x \leftrightarrow z) G(y,z) L^e(z \to y)}{p_l(y;z)}. \quad (14)$$

These estimates are in general not unbiased, because of the reasons discussed in section 3.3. We combine them with the unbiased estimates (13) by means of multiple importance sampling[24]: we make weighted sums of the estimates (13) with (14) for a number of pixels j. We choose the weights w_{ij} in the combination proportional to $p_r(o,x';y) g(x',y) p_l(y;z)$. The factors $\beta^j(x')$ and $\beta^i(x)$ cancel with $M^j(x')$ and $M^i(x)$ in (14).

The results of sections 3.2 and 3.3 can be derived in a similar way from integral L_i^2, which is identical to L_i^1 except that $L^e(z \to y)$ is substituted with $L^{ne}(z \to y)$. As the points z are now obtained by BSDF sampling at y (pdf $p_r(x',y;z)$), the combination weights can be taken proportional to $p_r(o,x';y) g(x',y) p_r(x',y;z)$.