Eurographics Symposium on Rendering 2009
Hendrik P. A. Lensch and Peter-Pike Sloan
(Guest Editors)

Volume 28 (2009), Number 4

Deterministic Importance Sampling with Error Diffusion

László Szirmay-Kalos and László Szécsi

Department of Control Engineering and Information Technology, Budapest University of Technology
Email: szirmay@iit.bme.hu

Abstract

This paper proposes a deterministic importance sampling algorithm that is based on the recognition that delta-sigma modulation *is equivalent to importance sampling. We propose a generalization for delta-sigma modulation in arbitrary dimensions, taking care of the curse of dimensionality as well. Unlike previous sampling techniques that transform low-discrepancy and highly stratified samples in the unit cube to the integration domain, our error diffusion sampler ensures the proper distribution and stratification directly in the integration domain. We also present applications, including environment mapping and global illumination rendering with virtual point sources.*

1. Introduction

In computer graphics we often need to evaluate high-dimensional integrals. Classical quadrature rules fail in higher dimensions due to the *curse of dimensionality*, which means that the sample number required for a given accuracy grows exponentially with the dimension. This problem can be avoided by Monte Carlo or quasi-Monte Carlo quadrature, which transforms random or quasi-random samples uniformly filling a high-dimensional unit cube to the integration domain. The error of the quadrature is smaller if the transformation mimics the integrand, i.e. it places more samples to regions where the integrand is large. Finding such a transformation is called *importance sampling*.

An alternative method of finding samples for integral quadratures is *adaptive sampling* that uses the samples to define a piece-wise constant approximation of the integrand, which is then analytically integrated. The error is small if the samples carry information where the integrand changes significantly. In contrast to important sampling that puts samples where the original integrand is large, adaptive sampling places samples where its derivative (gradient) is high.

In order to reduce the cost of dealing with the integrand during sampling and to allow vector valued functions where terms like "large" are meaningless, we define a scalar valued *importance function*, which should be cheaper to evaluate than the original function. The importance function mimics the integrand in case of importance sampling. For sample generation, we need the normalized version of the importance function, which is called the *target density*. The normalization factor is obtained as the integral of the importance function, possibly also by a numerical quadrature.

In this paper we propose a general high-dimensional sampling scheme that is called *error diffusion sampling* to highlight the similarities with error diffusion halftoning and delta-sigma modulation. The main novelty of the proposed sampling method is that it takes care of the proper distribution and stratification simultaneously and directly in the integration domain. We do not impose requirements on the importance function that would prohibit it to well mimic the integrand. In particular, unlike most of the previous approaches, we do not require the importance function to be analytically integrable with an invertible integral. The method is *consistent*, i.e. the error converges to zero if the number of samples goes to infinity. The paper is organized as follows. In Section 2 the previous work is surveyed in related fields like importance and adaptive sampling, and halftoning. Section 3 discusses delta-sigma modulation and Section 4 proposes the error diffusion sampler. Section 5 presents applications in the context of global illumination rendering.

2. Previous work

Uniform sample generation: The quality of uniform sample distributions can be described by their *discrepancy* [Shi91], which expresses the distance of the empirical cumulative distribution of the finite number of samples and the required theoretical uniform distribution. Random sam-

ples are distributed in a unit cube more uniformly than regular grids in D-dimensions if $D \geq 3$, i.e. their discrepancy is asymptotically $\mathcal{O}(\sqrt{\log\log N/N})$ while the discrepancy of a regular grid is $\mathcal{O}(N^{-1/D})$. *Low-discrepancy series* developed for quasi-Monte Carlo integration [Nie92], e.g. the Halton, Hammersly, Sobol, etc. series, or (t,m,s)-nets [Nie92, KK02] have better discrepancy $\mathcal{O}(\log^D N/N)$ than that of the random points. Poisson-disk distribution is known to have very good quality, but its generation involves many dart-throwing trials or lengthy iterative refinement [Coo86]. Recent papers proposed faster methods for Poisson-disk [DH06] or Poisson-disk-like [KCODL06, LD06, Ost07] distributions with the help of sophisticated tiling, but considered only 2D domains like a planar or spherical square.

Nonuniform sample generation: *Importance sampling* should place samples more densely where the integrand is large. The simplest approach is to find an analytical mapping from the integration domain to the unit cube such that important regions of the domain correspond to larger volumes in the unit cube. The problem with this approach is that it requires the analytical integration of the target density and its inversion during sample generation, thus, only relatively simple functions can be used. *Rejection sampling* can generate samples according to an arbitrary target density g by accepting or rejecting tentative samples proposed by another *source density* p. Rejection sampling requires a preferably small constant upperbound for ratio g/p and it may throw an unlimited number of samples away before it obtains a real sample. *Bi-directional importance sampling* [BGH05], *Importance resampling* [TCE05] or *Hierarchical thresholding* [ARBJ03, ODJ04, RCL+08] also generate samples with an easy source density, then the samples are randomly rejected and re-weighted to better mimic the target distribution. The *Metropolis method* [VG97] samples according to a target density that is the stationary distribution of an appropriately defined Markov chain. That is, it only converges to the target density, thus it suffers from initial bias. The *Population Monte Carlo* [GmMR04, LFCD07] approach mutates a set of samples to get the sample set in the next iteration and weights them to compensate that they were not drawn from the target distribution.

Sample relaxation: Importance sampling algorithms discussed so far ignore the problem that the samples are not well stratified, i.e. the empirical distribution of their finite number of samples may be far from the prescribed distribution. Even if they start from a low-discrepancy series, the transformation distorts the original distribution, thus its stratification is corrupted. To address this problem in the context of environment mapping, Kollig used an iterative relaxation algorithm [KK03], which can be speeded up by exploiting a tiling structure [ODJ04]. Wan et al. [WWL05] proposed sampling directly on the sphere. Agarwal et al. [ARBJ03] ensured stratification by an additional partitioning step executed as post-processing after sample generation. Recently, Spencer [BS09] proposed relaxation to post-process hit samples in a photon map. Unfortunately, all these techniques are practical only in 2D.

Adaptive sampling: Adaptive sampling uses samples to build a piece-wise constant approximation of the integrand, thus it is accurate if samples are concentrated where the integrand changes significantly. Image space adaptive sampling was used even in the first ray tracer [Whi80], and has been improved by randomization [BM97] and by the application of information theory tools [RFS03]. Adaptive sampling methods should decompose the sampling domain, which is straightforward in 2D but needs special data structures like the kd-tree in higher dimensions [HJW+08].

Halftoning: Halftoning renders gray-scale images on a bi-level display, placing more white points at brighter areas and fewer points at darker parts to make the spatial density of white points in a region around a pixel proportional to the gray level of that particular pixel. If we consider the gray level of the original image to be the importance function and the white pixels of the resulting image to be sample locations, then halftoning becomes equivalent to importance sampling. This holds for an arbitrary halftoning algorithm, including the random and ordered halftoning methods, and the error diffusion halftoning as well. In fact, error diffusion halftoning is the generalization of the delta-sigma modulation to 2D [KEBS97]. Algorithms of error diffusion halftoning, including the Floyd-Steinberg [FS75], Jarvis [JJN76], and Stucki [Stu81] methods, differ in the error shaping filter and in the order they visit the image pixels. We have already exploited the similarity of halftoning and importance sampling in environment mapping in [SKSP09]. Now, we step forward and show that this similarity has important theoretical reasons and roots in delta-sigma modulation. Furthermore, the basic idea is generalized to higher dimensions as well. Finally, new application possibilities are presented not only in environment mapping but also in virtual point light source based global illumination rendering.

3. Delta-sigma modulation

For the sake of notational simplicity, let us consider first a 1D integral of scalar valued integrand $f(t)$ in the domain of $[0,1]$ (notations are summarized in Table 1). Let us find importance function $I(t)$ that mimics $f(t)$, and define the *cumulative importance* $b(t)$ as the integral of the importance:

$$b(t) = \int_0^t I(\tau)\mathrm{d}\tau.$$

The target density $g(t) = I(t)/b_T$ is the normalization of the importance function by the total cumulative importance $b_T = b(1)$. The ratio of the integrand and the target density, i.e. the part that is not mimicked, is denoted by $h(t) = f(t)/g(t)$. Suppose that samples $t_1, t_2, \ldots, t_M$ are obtained in an *arbitrary* way and the integral is approximated by the

Notation	Meaning
t	sample in unit interval $[0,1]$
$\mathbf{u}$	sample in D-dimensional unit cube $\mathcal{U}$
$\mathbf{z}$	sample in D-dimensional target domain $\mathcal{P}$
$f(.)$	integrand
$I(.)$	importance function that mimics integrand f
$b(.)$	cumulative importance: $b(t)=\int^t I(\tau)\mathrm{d}\tau$
b_T	total importance
$\tilde{b}_T$	approximated total importance
$g(.)$	target density: $g(t)=I(t)/b_T$
$h(.)$	integrand per target density: $h(t)=f(t)/g(t)$
M	number of real samples
N	number of tentative samples
$m(t)$	number of samples smaller than t
$p(.)$	source density

Table 1: *Notations of the paper.*

usual quadrature:

$$\int_0^1 f(t)\mathrm{d}t \approx \frac{1}{M}\sum_{j=1}^{M}\frac{f(t_j)}{g(t_j)} = \frac{b_T}{M}\sum_{j=1}^{M}\frac{f(t_j)}{I(t_j)}. \qquad (1)$$

The error of the quadrature can be expressed as (for details see the appendix):

$$\left|\int_0^1 f(t)\mathrm{d}t - \frac{1}{M}\sum_{j=1}^{M}\frac{f(t_j)}{g(t_j)}\right| = \left|\int_0^1 h'(t)\left(\frac{b(t)}{b_T}-\frac{m(t)}{M}\right)\mathrm{d}t\right|, \qquad (2)$$

where $h'(t)$ is the derivative of $h(t)$ and $m(t)$ is the number of sample points that are smaller than t. This error formula shows two ways to reduce the error.

The first approach would reduce factor $|h'(t)|$, that is, it would make $h(t)=f(t)/g(t)$ close to constant, or alternatively, target density $g(t)$ approximately proportional to integrand $f(t)$, which is the objective of importance sampling.

The second approach would reduce factor

$$D(t)=\left|\frac{b(t)}{b_T}-\frac{m(t)}{M}\right|$$

which expresses how well samples $t_1,t_2,\ldots,t_M$ follow the target density $g(t)$. If the samples were allocated exactly according to the target density, then the relative number of samples in $[0,t]$ would be $b(t)/b_T$. However, the real relative number of samples in $[0,t]$ is $m(t)/M$. Low-discrepancy sampling and adaptive sampling focus on the reduction of this factor.

As in practical cases none of the two factors can be made constant zero, a good sampling strategy should simultaneously minimize both of them, i.e. it should both make the target density mimic the integrand and generate finite number of samples so that they well follow the distribution prescribed by the target density. The primary objective of this paper is to propose a sampling scheme that simultaneously considers both factors. In this sense, our scheme can also be interpreted as a combination of the advantages of importance sampling and adaptive sampling.

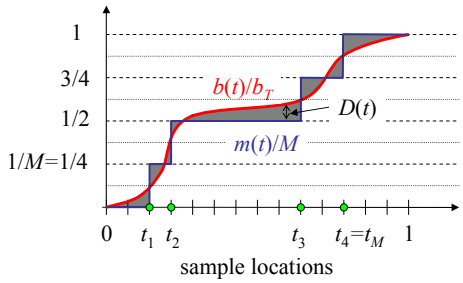

Figure 1: *The interpretation of error factor $D(t)$ that is the difference of the relative cumulative importance $b(t)/b_T$ and the empirical cumulative distribution $m(t)/M$ (here $M=4$).*

Function $m(t)$ counting the samples smaller than t is integer valued and increments at sample locations, thus the $m(t)/M$ term makes jumps of size $1/M$ (Figure 1). On the other hand, the cumulative importance is continuous, thus $D(t)$ cannot be reduced under $1/(2M)$ everywhere. However, this optimum can be reached in 1D in the following way. The importance is integrated increasing t in $b(t)$ and we continuously check whether it exceeds $1/(2M)$. This comparison is implemented inputting the difference to a 1-bit *quantizer*. If the difference gets greater than $1/(2M)$, a sample is generated and the original integrand is evaluated here. At the same time, the running integral of $b(t)$ is reduced by $1/M$, and the same procedure continues. The process is well-known in signal processing and is called *delta-sigma modulation* [DeF74]. The described implementation has two drawbacks. It has a delay, thus the sample is generated after an important region. On the other hand, the noise of quantization (i.e. the error of sampling the original function only at "rare" discrete points) is directly added to the output. Therefore, in practice, another version, called the *Noise-Shaping Feedback Coder* (NSFC) is preferred.

NFSC is discussed switching from continuous time analysis to discrete time, assuming that the $[0,1]$ interval is decomposed to N small intervals of size $\Delta t=1/N$ and using the $t=i\Delta t$ substitution. NSFC (left of Figure 2) computes error variable $e(i)$ as the difference of the preferred output and the output of the quantizer. This error is added to the subsequent samples, i.e. the error is integrated, to lift the next sample over the quantization level.

In discrete temporal domain, the state equations for input $x(i)$, quantizer's input $q(i)$, output $y(i)$, and error $e(i)$ are:

$$q(i)=x(i)+\mathcal{H}(e(i)),\quad e(i)=q(i)-y(i),\quad y(i)=\mathcal{Q}(q(i)),$$

where $\mathcal{H}$ is the response of the noise shaping error filter and $\mathcal{Q}$ is the response of the quantizer. Unfortunately, the

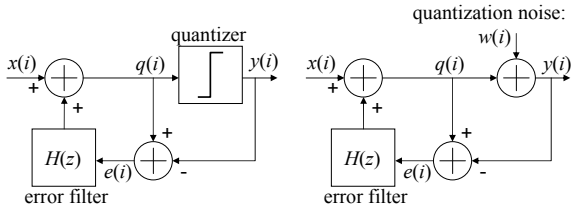

Figure 2: *Noise shaping feedback coder (right) and the linear model of the same network that replaces the non-linear quantizer by adding white noise (right).*

quantizer is a non-linear element, which forbids transform domain analysis. The usual trick is to model the effect of the quantizer as the addition of *white noise* $w(i)$ [KEBS97] (right of Figure 2). Transforming the state equations from the discrete temporal domain to the z-domain, we obtain

$$Y(z) = X(z) + W(z)(1 - H(z)),$$

where Y, X, and W are the z-transforms of y, x, and w, respectively, and $H(z)$ is the transfer function of the error filter. Note that input x shows up in the output without scaling and delay. On the other hand, the white noise added by the quantizer is shaped. If $H(z)$ is a low-pass filter, then $(1 - H(z))$ will be high-pass, thus white noise is turned to *blue noise*, i.e. the sample points will be *stratified*.

The discussed coder can directly be used in 1D importance sampling. The domain is scanned two times in small steps $\Delta t = 1/N$. The visited points are called the *tentative samples* to emphasize that the real sample points will be among these but not all tentative samples will act as a real one. In the first run, the importance function is evaluated at the tentative samples and its integral is approximated as

$$b_T = \int_0^1 I(\tau)\mathrm{d}\tau \approx \sum_{i=0}^{N} I(i\Delta t)\Delta t = \tilde{b}_T. \qquad (3)$$

Then, in the second run, the same importance values are taken at the visited points and input to the NFSC, i.e. $x(i) = I(i\Delta t)$. If the quantizer comparing to $1/(2M)$ flops, i.e. when $q(i)$ reaches $1/(2M)$ and thus $y(i)$ changes from zero to $1/M$, the original integrand f is evaluated and its value is included into the integral quadrature of equation (1). Simultaneously, the error value is obtained as the difference of the quantizer's input q and the current output $1/M$. If the quantizer does not flop, the difference is purely the quantizer's input.

The output of the noise shaping filter is added to the subsequent values, i.e. the error of the current sample is distributed among them. This filter is low pass if the weights are all non negative. For example, a low pass filter with transfer $H(z) = 0.5z^{-1} + 0.5z^{-2}$ would add half of the error to the next sample, and another half to the sample after the next one. We usually require that the weights are not only positive (low-pass criterium) but also that their sum is 1. It means that the noise transfer function has a zero at zero frequency (DC), thus the average of the signal is unchanged.

In order to make this sampling scheme feasible for rendering applications, we should extend it to (arbitrarily) high-dimensions. The extension requires an *order* of the tentative sample points in which they are visited. Note that unlike in other Monte Carlo or quasi-Monte Carlo methods where sample points can be processed independently and in any order, here the order is important. Solutions like defining the tentative samples as points of a regular grid and exploring the high-dimensional space row-by-row or following a space filling Peano or Hilbert curve suffer from the *curse of dimensionality*. Thus, these methods may be good in 2D, but become prohibitively expensive in higher dimensions, where we need another approach. Furthermore, we also need the *neighborhood* of a tentative sample where its error can be distributed. The neighborhood may contain only unvisited samples, otherwise the system would have to process the samples multiple times.

4. The error diffusion sampler

Let us consider the integral of a possibly vector valued function $f(\mathbf{z})$ in a D-dimensional domain $\mathcal{P}$. First, the general domain is mapped to a D-dimensional unit cube $\mathcal{U}$ using *source density* $p(\mathbf{z})$ that must be analytically integrable in $\mathcal{P}$ and its integral must be invertible. The source density is interpreted as the Jacobi determinant of the mapping:

$$\mathrm{d}\mathbf{u}(\mathbf{z}) = p(\mathbf{z})\mathrm{d}\mathbf{z}, \qquad p(\mathbf{z}) = \left|\frac{\mathrm{d}\mathbf{u}(\mathbf{z})}{\mathrm{d}\mathbf{z}}\right|.$$

The original integral is written as follows:

$$Q = \int_{\mathcal{P}} f(\mathbf{z})\mathrm{d}\mathbf{z} = \int_{\mathcal{P}} \frac{f(\mathbf{z})}{p(\mathbf{z})} p(\mathbf{z})\mathrm{d}\mathbf{z} = \int_{\mathcal{U}} \frac{f(\mathbf{z}(\mathbf{u}))}{p(\mathbf{z}(\mathbf{u}))}\mathrm{d}\mathbf{u}, \qquad (4)$$

where $\mathbf{z}(\mathbf{u})$ is the inverse of $\mathbf{u}(\mathbf{z})$.

Importance function $I(\mathbf{u})$ is defined in the unit cube and it should mimic the transformed integrand f/p. In fact, we expect the error diffusion sampler to compensate those factors that could not be compensated by the application of source density p. Target density $g(\mathbf{u})$ is the normalized version of the importance function.

In order to explore the integrand in the D-dimensional unit cube $\mathcal{U}$ and consequently in domain $\mathcal{P}$, we take a low-discrepancy series and generate N *tentative sample points* $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ with it. We work with the Halton series, but other low discrepancy series could be used as well. The tentative sample points are visited two times. First, during their generation, we follow the *generation order* of the low-discrepancy series constructing method, and estimate the to-

tal importance:

$$b_T = \int_{\mathcal{U}} I(\mathbf{u})\mathrm{d}\mathbf{u} \approx \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{u}_i) = \tilde{b}_T.$$

The computed importance values are stored as initial values for the *errors* that are diffused in the second phase.

In the second phase, the tentative samples are scanned again and M real samples are generated for the integral quadrature Q. We emphasize that now we do not follow the generation order but step from a tentative sample point to its *neighbor* being in its vicinity. Formally, now in step j, we use the sample generated as sample $i = O(j)$, where function O is responsible for ordering according to the neighborhood relation. At sample point $\mathbf{u}_{O(j)}$ the error is compared to the threshold $\tilde{b}_T/(2M)$. The comparison may have two results:

- If the error is greater than the threshold, then integrand f is evaluated at tentative sample $\mathbf{u}_{O(j)}$ and is included in the quadrature of Q:

$$Q \mathrel{+}= \frac{\tilde{b}_T}{M}\frac{f(\mathbf{z}(\mathbf{u}_{O(j)}))}{p(\mathbf{z}(\mathbf{u}_{O(j)}))I(\mathbf{u}_{O(j)})}.$$

  Simultaneously, the error of the current tentative sample is decreased by $\tilde{b}_T/M$.
- If the error is not greater than the threshold, then no sample is generated here and the error of the tentative sample is left unchanged.

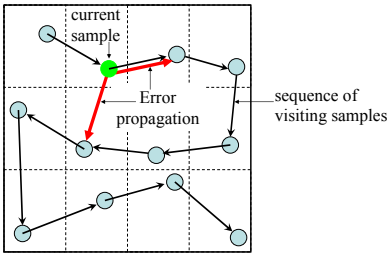

Figure 3: *The sequence of tentative samples and the error distribution to unvisited neighbors in $D = 2$ dimensions.*

In both cases, before stepping onto the next neighboring tentative sample point, the remaining error of the current tentative sample point is distributed to its unvisited neighbors. Finally, an unvisited neighbor is selected and the method continues similarly.

A critical issue in this algorithm is how to find the "neighboring" points in a low-discrepancy series. The issue is targeted by the following subsection.

4.1. Neighborhood relation in low-discrepancy series

In order to develop neighborhood relation, we have to understand how low-discrepancy series fill the unit cube. In particular, we take the Halton series, but other series behave similarly in this respect.

The construction of the 1D Halton series expresses the index i in base B and finds the radical inverse that mirrors the number onto the "decimal" point to obtain the coordinate of the sample in $(0,1)$. Thus, as i is incremented from 1 to B^R, then even further toward B^{R+1}, the construction algorithm generates all R-long combinations of digits $0,1,\ldots,B-1$ before producing a combination of length $R+1$. This means that after the radical inverse, the sequence will visit all intervals of length $[kB^{-R},(k+1)B^{-R})$ before putting a new point in an interval already visited. This is often called the *elemental interval property* [KK02, DBMS02]. As i increases, the algorithm produces a single point in each interval of length B^{-1}, then in each interval of length B^{-2}, etc.

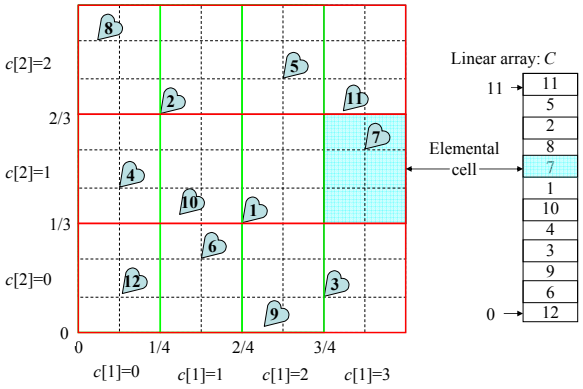

Figure 4: *The first $N = 12$ points of a 2D Halton series of bases $B_1 = 2$ and $B_2 = 3$. Setting $R_1 = 2$ and $R_2 = 1$ meets the requirement that $N = B_1^{R_1}B_2^{R_2} = 2^2 3^1$ and that the cells are cube like since $B_1^{R_1} = 4 \approx B_2^{R_2} = 3 \approx N^{1/D} = 3.46$. Note that the elemental cells have area $B_1^{-R_1}B_2^{-R_2} = 1/12$ and each cell contains exactly one point.*

In 2D, the generation algorithm would visit all columns of width $B_1^{-R_1}$ before visiting a column again, and similarly it would visit all rows of height $B_2^{-R_2}$ before putting a new point into a row (Figure 4). The columns and rows form $B_1^{R_1}B_2^{R_2}$ cells. Since the periodicity of the columns and rows are $B_1^{R_1}$ and $B_2^{R_2}$, respectively, the periodicity of the cells is the smallest common multiple of $B_1^{R_1}$ and $B_2^{R_2}$. If B_1 and B_2 are relative primes, then this equals to the their product, that is total number of cells.

In D-dimensions, N points of the Halton sequence of bases $B_1,\ldots,B_D$ will implicitly induce a cell structure of resolution $B_1^{R_1} \times \ldots \times B_D^{R_D}$, where each cell contains at most one point if $B_1^{R_1}B_2^{R_2}\ldots B_D^{R_D} \leq N$ holds. All cells have the same volume $B_1^{-R_1}B_2^{-R_2}\ldots B_D^{-R_D}$. If the number of samples N is equal to $B_1^{R_1}B_2^{R_2}\ldots B_D^{R_D}$, then every cell will contain exactly one point from the generated N. As we shall assume that neighboring cells are at approximately the same distance, the resolution along an axis should be found to

make the cells cube-like and having similarly long edges, i.e. $B_1^{R_1} \approx B_2^{R_2} \approx \ldots \approx B_D^{R_D} \approx N^{1/D}$.

Our error diffusion algorithm will use this induced cell structure for finding neighboring tentative samples.

4.2. Datastructure and the algorithm

The induced grid is a D-dimensional array that is indexed by row, column, etc. numbers that are collected in vector $\mathbf{c} = [c[1], c[2], \ldots c[D]]$. The high-dimensional array is linearized and stored in a 1D array C indexed by

$$j = c[1] + c[2]B_1^{R_1} + c[3]B_1^{R_1}B_2^{R_2} + \ldots + c[D]B_1^{R_1}\ldots B_{D-1}^{R_{D-1}}.$$

An array element stores the generation index of the low-discrepancy point that is contained by the corresponding cell ($C[j].i$), the importance ($C[j].I$), and the error value of this point ($C[j].e$) that is initialized to its importance. The determination of the containing cell from the coordinates of the point is based on the fact that along coordinate d, cell boundaries are at the integer multiples of $B_d^{-R_d}$. Thus a point of coordinates $\mathbf{u} = [u[1], \ldots u[D]]$ is included in cell $[c[1], c[2], \ldots c[D]]$ where $c[d]$ is the integer part of $u[d]B_d^{R_d}$.

In the first phase of the algorithm, low-discrepancy points are generated with the radical inverse function Φ_B, so cells are visited one-by-one calculating the importance of the point and summing it to get the total importance $\tilde{b}_T$:

$\tilde{b}_T = 0$; *// normalization constant*
for $i = 1$ **to** N **do** *// generate N tentative samples*
  for $d = 1$ **to** D **do**
    $u[d] = \Phi_{B_d}(i)$; *// coordinate*
    $c[d] = (\text{int})(u[d]B_d^{R_d})$; *// cell index*
  endfor
  $C(\mathbf{c}).I = I(\mathbf{u})$; *// importance of the tentative sample*
  $C(\mathbf{c}).i = i$; *// store generation index*
  $C(\mathbf{c}).e = C(\mathbf{c}).I$; *// initialize the error with importance*
  $\tilde{b}_T$ += $C(\mathbf{c}).I$; *// accumulate the normalization constant*
endfor

In the second phase, the cells are visited row by row, and the integral quadrature is computed in variable Q:

$Q = 0$; *// integral estimate*
for $j = 0$ **to** $N-1$ **do** *// scan tentative samples again*
  $j_s = j$;
  for $d = 1$ **to** D **do** *// find cell indices* $\mathbf{c} = [c[1], \ldots, c[D]]$
    $c[d] = j_s \bmod B_d^{R_d}$;
    $j_s = j_s \operatorname{div} B_d^{R_d}$;
  endfor
  if $(C[j].e \geq \tilde{b}_T/(2M))$ *// quantization*
    $C[j].e$ -= $\tilde{b}_T/M$; *// real sample*
    for $d = 1$ **to** D **do** $u[d] = \Phi_{B_d}(C[j].i)$; *// coordinates*
    Q += $\tilde{b}_T/M \cdot f(\mathbf{z}(\mathbf{u}))/p(\mathbf{z}(\mathbf{u}))/C[j].I$; *// quadrature*
  endif
  // distribute error in neighbors
  for $d = 1$ **to** D **do** $C(\mathbf{c} + \Delta_d).e$ += $C[j].e \cdot w_d$;
endfor

The error shaping filter is defined by index offsets Δ_d selecting neighboring cells and weights w_d. Our error distribution uses index offset Δ_d that is a D-dimensional vector of all zeros but at position d where it has 1. With this neighborhood definition a cell has D unvisited neighboring cells, which share a face with the current cell. Conveniently, the weighting factors w_d can all be $1/D$, but according to our experience, the quality of the sample distribution can be improved by giving larger weights to faster changing cell coordinates. Other neighborhoods of cells sharing an edge or a vertex may also result in better error shaping, but the number of neighbors would grow rapidly in higher dimensions.

5. Applications

To apply the proposed sampling scheme, we first have to formulate our problem as a multi-dimensional integral and map the integration domain to the unit cube by a source density. Then, the importance function has to be defined. The importance function should accurately follow the ratio of the integrand and the source density, but it should be much cheaper to evaluate since otherwise the overhead of computing the importance of tentative samples would be too high.

5.1. Environment mapping

Environment mapping [Deb98] computes the reflected radiance of point $\vec{x}$ as

$$L(\vec{x}, \vec{\omega}) = \int_{\Omega} L^{env}(\vec{\omega}') f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cos\theta' v(\vec{x}, \vec{\omega}') \mathrm{d}\omega',$$

where Ω is the set of all directions, $L^{env}(\vec{\omega}')$ is the radiance of the environment map at direction $\vec{\omega}'$, f_r is the BRDF, and $v(\vec{x}, \vec{\omega}')$ is the indicator function checking whether no virtual object is seen from $\vec{x}$ at direction $\vec{\omega}'$. Note that the integral is the product of three factors, the light intensity, the cosine weighted BRDF, and the visibility indicator:

$$f(\vec{\omega}') = L^{env}(\vec{\omega}') f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cos\theta' v(\vec{x}, \vec{\omega}').$$

Now we have to decide which factors are mimicked by the source density and the importance function, respectively. Two options are investigated, light source sampling and product sampling.

Light source sampling: In this case, the source density is independent of the scene and simply maps directional domain Ω to the unit square. The importance function mimics the environment illumination $L^{env}(\vec{\omega}')$ weighted by solid angle $\Delta\omega$ corresponding to the area of this particular texel. Weights $\Delta\omega$ are not constant and depend on the environment map parametrization. Ignoring the BRDF, factor $\cos\theta'$, and the visibility degrades importance sampling, but, as the importance function depends just on the direction, tentative samples need to be generated only once for all pixels.

The environment illumination that should be mimicked by

the importance function is available as a 2D texture map, where texels are the tentative samples. Note that in this case, the proposed error diffusion scheme is as simple as the Floyd-Steinberg halftoning of the weighted environment map. Figures 5 and 6 compare the distribution of samples and the resulting images of the illuminated object for random sampling and for the error diffusion sampler. As the error diffusion sampler scans the map only once and obtains samples directly, it is not slower than random sampling. Sampling the 1024×512 resolution environment map of Figure 5 takes 47 msec both with random sampling and with error diffusion on an nVidia GeForce 8800 GFX GPU. After sampling, rendering with shadow mapping requires 1.2 sec.

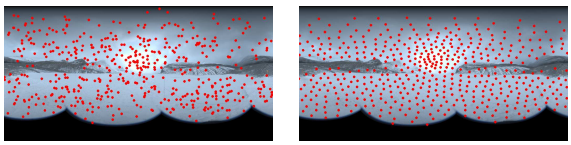

Figure 5: *Sampling weighted environment maps with random sampling (left) and error diffusion (right).*

Random sampling Error diffusion Reference

Figure 6: *Diffuse and specular bunnies illuminated by directional lights sampled randomly and with error diffusion.*

Product sampling: Suppose that we use BRDF sampling as a source density, which compensates the angular variation of the BRDF and the $\cos\theta'$ factor, and replaces them by the albedo a:

$$\frac{f(\vec{\omega}')}{p(\vec{\omega}')} = L^{env}(\vec{\omega}')a(\vec{x},\vec{\omega})v(\vec{x},\vec{\omega}').$$

Note that the source density could not compensate the environment radiance and the visibility function.

The computation of the importance must be much cheaper than that of the original integrand f. In environment mapping, the expensive part is the visibility test, so we ignore occlusions in the importance. The environment illumination and the albedo are wavelength dependent, thus the importance should map these vectors to scalars. We use the luminance ($\mathcal{L}$) of the product. Thus, the importance function is

$$I(\mathbf{u}) = \mathcal{L}(L^{env}(\vec{\omega}'(\mathbf{u})a(\vec{x},\vec{\omega})).$$

The terms inserted into the integral quadrature are

$$\frac{\tilde{b}_T}{M}\frac{f(\mathbf{z}(\mathbf{u}_j))}{p(\mathbf{z}(\mathbf{u}_j))I(\mathbf{u}_j)} = \frac{\tilde{b}_T}{M}\frac{L^{env}(\vec{\omega}'(\mathbf{u}_j))a(\vec{x},\vec{\omega})v(\vec{x},\vec{\omega}'(\mathbf{u}_j))}{\mathcal{L}(L^{env}(\vec{\omega}'(\mathbf{u}_j))a(\vec{x},\vec{\omega}))}.$$

Note that this is the best importance sampling provided that the visibility is not included in the importance and both the environment lighting intensity and the BRDF are spectra.

In order to test the approach, we have compared three techniques: BRDF sampling, importance resampling [TCE05], and the new error diffusion scheme. All three were implemented as GPU algorithms, which run on nVidia GeForce 8800 GFX graphics hardware. All methods traced $M = 32$ rays per pixel. Both importance resampling and the error diffusion sampler generated the real samples from $N = 1024$ tentative samples obtained without ray tracing. The results are shown by Figure 7. Note that the error diffusion sampler completely eliminated the noise at fully visible surfaces and some noise remained only at partially occluded parts. We observed that the sample generation overhead becomes negligible in comparison to sample evaluation as soon as the scene consists of more than 10K triangles.

5.2. Rendering with virtual point lights

The virtual light source [Kel97, WKB+02, WFA+05] method is a bi-directional global illumination algorithm, where shooting walks are generated and their hit points act as virtual point-like light sources (*VPL*) providing the indirect illumination at the points visible from the camera. If all surfaces are diffuse, the estimator for the radiance of point $\vec{x}$ due to a single VPL of radiant exitance Φ at point $\vec{y}$ is

$$L(\Phi(\vec{y}) \to \vec{x}) = \Phi(\vec{y})G(\vec{x},\vec{y})v(\vec{x},\vec{y})f_r(\vec{x}), \qquad (5)$$

where $v(\vec{x},\vec{y})$ is 1 if $\vec{x}$ and $\vec{y}$ are visible from each other and zero otherwise, and $G(\vec{x},\vec{y})$ is the geometry factor:

$$G(\vec{x},\vec{y}) = \frac{\cos\theta_{\vec{y}}\cos\theta_{\vec{x}}}{\pi|\vec{x}-\vec{y}|^2}.$$

In the geometry factor $\theta_{\vec{x}}$ and $\theta_{\vec{y}}$ are the angles between the surface normals and the connection direction at $\vec{x}$ and $\vec{y}$.

Denoting the position of the kth hit point of the ith shooting walk by $\vec{y}_i^k$, the total contribution of all VPLs to the radiance of point $\vec{x}$ is

$$L(\vec{x}) = \sum_i \sum_k \Phi(\vec{y}_i^k)G(\vec{x},\vec{y}_i^k)v(\vec{x},\vec{y}_i^k)f_r(\vec{x}). \qquad (6)$$

The weak points of the estimator are revealed by this formula. The geometry factor is not compensated by the sampling density used to generate shooting walks. The distance

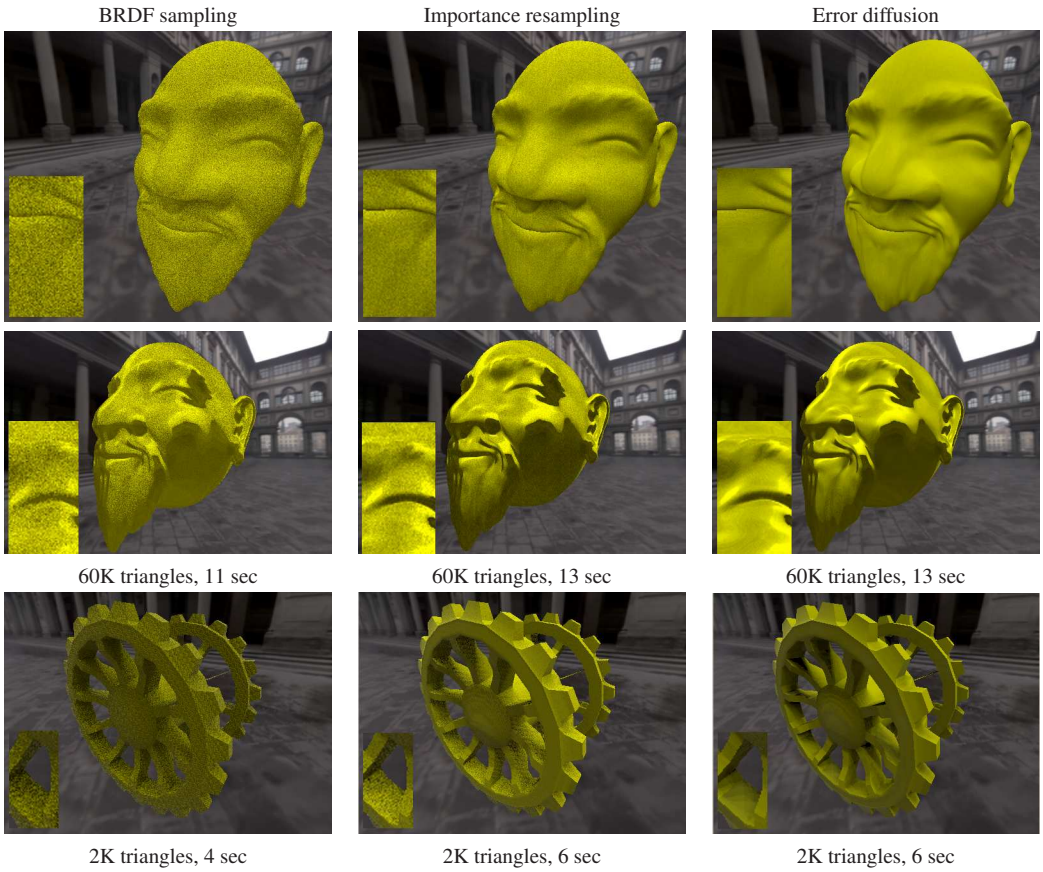

Figure 7: *Environment map sampling. Note that the error diffusion sampler eliminated the noise at fully visible surfaces both for the diffuse and specular cases. The lower row of images show a wheel having a lot of occlusions.*

between the VPL and the illuminated point can be arbitrarily small, which introduces high variation. As a result of this bad sampling, bright spots show up in the corners of the scene.

We apply the error diffusion sampler to solve this problem. The original shooting algorithm is responsible for the source density, which generates N shooting paths from unit cube points $\mathbf{u}_1, \ldots, \mathbf{u}_N$. Assuming that the real source of light is a large area light and generating 1 ray long shooting paths, the sample domain is 4D. For point $\vec{x}$ and sample $\mathbf{u}_i$, we define the importance function as the luminance of the reflected radiance of virtual point sources in the shooting path defined by $\mathbf{u}_i$, replacing the original visibility indicator v by an approximate indicator V. A conservative approximation of the visibility term between point $\vec{x}$ and a VPL can be obtained by checking the occlusion with a proxy geometry, for example, a collection of spheres contained by the original objects. The resulting importance function is

$$I(\mathbf{u}_i, \vec{x}) = \sum_k \mathcal{L}(\Phi(\vec{y}_i^k) f_r(\vec{x})) G(\vec{x}, \vec{y}_i^k) V(\vec{x}, \vec{y}_i^k).$$

For each $\vec{x}$, the error diffusion sampler selects M shooting paths associated with primary samples $\mathbf{u}_{i_1}, \ldots, \mathbf{u}_{i_M}$ from the N tentative paths, and results in the following estimator

$$L(\vec{x}) = \frac{\tilde{b}_T(\vec{x})}{M} \sum_{j=1}^{M} \sum_k \frac{\Phi(\vec{y}_{i_j}^k) f_r(\vec{x}) G(\vec{x}, \vec{y}_{i_j}^k) v(\vec{x}, \vec{y}_{i_j}^k)}{\sum_{k'} \mathcal{L}(\Phi(\vec{y}_{i_j}^{k'}) f_r(\vec{x})) G(\vec{x}, \vec{y}_{i_j}^{k'}) V(\vec{x}, \vec{y}_{i_j}^{k'})}.$$

Total importance $\tilde{b}_T(\vec{x})$ is accurate since it is a quadrature of a low-variation integrand and computed from all tentative samples. On the other hand, the high variation caused by the geometry term G has been greatly reduced by the denominator storing the weighted sum of the geometry terms associated with this path.

Concerning the cost, if the length of the shooting path is L and the number of pixels is P, then we trace NL rays to generate tentative shooting paths, and $P(L+1)M$ rays during final gathering. As we trace much more rays during final gathering than during shooting, the number of additional rays to be traced for not used tentative samples is negligible. We rendered the images in the upper row of Figure 8 taking

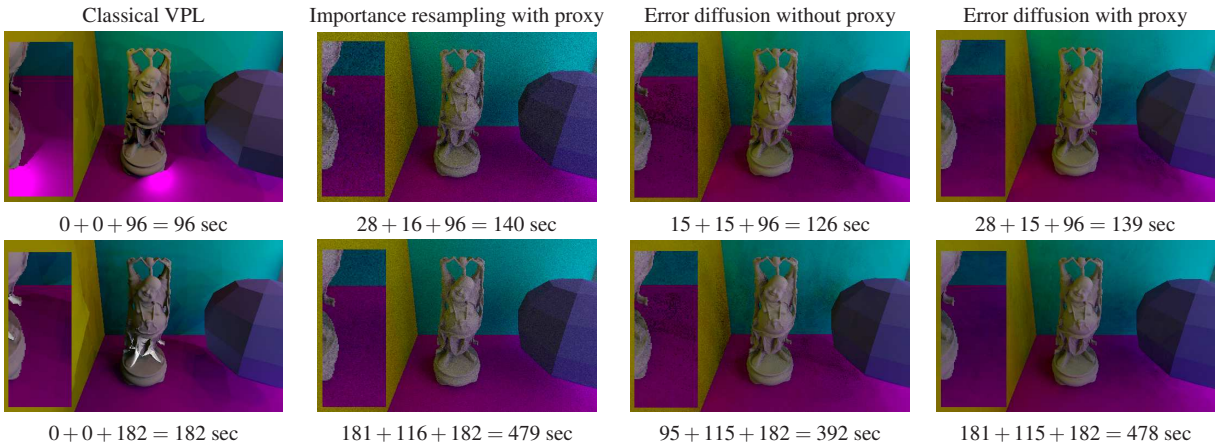

Figure 8: *Comparison of the classical VPL method, importance resampling with approximate visibility computed with a proxy geometry, and the error diffusion algorithm without and with approximate visibility. The proxy geometry is one sphere each for the faceted ball and for the Buddha model. Images in the upper row were rendered with with $M = 16$ samples per pixel selected from $N = 420$ global tentative samples, while the lower row was computed taking $M = 32$ samples per pixel selected from $N = 2978$ tentative samples. Computation times are measured on a CPU ray tracer and expressed as tentative sample generation + real sample generation + ray tracing.*

$M = 16$ VPLs in each pixel with the classical algorithm, with importance resampling, and with the error diffusion sampler. Final gathering traced $P(L+1)M = 8$ million rays. Importance resampling and the error diffusion sampler selected the VPLs from $N = 420$ tentative samples. In the lower row, we repeated the same experiment taking $M = 32$ real VPLs from $N = 2978$ tentative ones.

Figure 9: *Equal time comparison of the classical VPL method and the error diffusion algorithm with proxy.*

Figure 9 compares error diffusion sampling to the classical VPL method generating 3 ray long light paths started at an area light source (8D integration). Both algorithms run 520 seconds. We increased the albedos to emphasize longer light paths.

6. Conclusions

This paper presented a new importance sampling strategy that is based on delta-sigma modulation. The algorithm consists of two steps. In the first phase, tentative samples are generated transforming the Halton sequence with an easy source density. Then, only a subset of these tentative samples are used as real samples. The real sample locations are selected with an error diffusion halftoning like algorithm that has been generalized to arbitrary dimensions. The power of the method comes from the stratification enforced in the transformed domain, which is automatically provided by the error diffusion. The method is really effective when neighboring tentative samples are not very far from each other, i.e. when we can afford to generate many tentative samples. In addition to the discussed examples, possible future applications include depth of field, motion blur, single-scattering of complex lighting in participating media, etc.

Acknowledgement

This work has been supported by the National Office for Research and Technology and OTKA K-719922 (Hungary).

Appendix

In this appendix we rewrite the quadrature error. Let us first consider the original integral and apply partial integration using $b'(t) = I(t)$:

$$\int_0^1 f(t)\mathrm{d}t = \int_0^1 h(t)g(t)\mathrm{d}t = \int_0^1 h(t)\frac{I(t)}{b_T}\mathrm{d}t =$$

$$\left[h(t)\frac{b(t)}{b_T}\right]_0^1 - \int_0^1 h'(t)\frac{b(t)}{b_T}\mathrm{d}t = h(1) - \int_0^1 h'(t)\frac{b(t)}{b_T}\mathrm{d}t. \quad (7)$$

Now let us consider the $1/M\sum_{j=1}^{M} f(t_j)/g(t_j)$ term, express first $f(\tau)/g(\tau) = h(\tau)$ as the integral of its derivative, and extend the integration domain to $[0,1]$ by multiplying the integrand by a step function $\varepsilon(t-\tau)$:

$$h(\tau) = h(1) - \int_\tau^1 h'(t)\mathrm{d}t = h(1) - \int_0^1 h'(t)\varepsilon(t-\tau)\mathrm{d}t.$$

Substituting $t_1, t_2, \ldots, t_M$ into τ and computing the average, we get:

$$\frac{1}{M}\sum_{j=1}^{M} h(t_j) = h(1) - \int_0^1 h'(t)\frac{1}{M}\sum_{j=1}^{M}\varepsilon(t-t_j)\mathrm{d}t. \quad (8)$$

Note that $m(t) = \frac{1}{M}\sum_{j=1}^{M}\varepsilon(t-t_j)$ is the average number of samples that are smaller than t. Subtracting equation (8) from equation (7) we obtain equation (2).

References

[ARBJ03] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen. Structured importance sampling of environment maps. *ACM Trans. Graph.*, 22(3):605–612, 2003.

[BM97] M. R. Bolin and G. W. Meyer. An error metric for Monte Carlo ray tracing. In *Rendering Techniques '97*, pages 57–68, 1997.

[BGH05] D. Burke, A. Ghosh, and W. Heidrich. Bidirectional importance sampling for direct illumination. In *EG Symp. on Rendering*, pages 147–156, 2005.

[BS09] B. Spencer and M. W. Jones. Into the blue: Better caustics through photon relaxation. *Computer Graphics Forum*, 28(2):319–328, 2009.

[Coo86] R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986.

[DBMS02] K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel. Interactive global illumination using selective photon tracing. In *13th EG Workshop on Rendering*, pages 25–36, 2002.

[Deb98] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98 Proceedings*, pages 189–198, 1998.

[DeF74] R. DeFreitas. The low-cost way to send digital data: Delta-sigma modulation. *Electronic Design*, 22:68–74, 1974.

[DH06] D. Dunbar and G. Humphreys. A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. Graph.*, 25(3):503–508, 2006.

[FS75] R. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. In *Society for Information Display 1975 Symp. Digest of Tecnical Papers*, page 36, 1975.

[GmMR04] A. Guillin, J. M. Marin, and C. P. Robert. Population Monte Carlo. In *Comp. and Graph. Stat.*, 13:907–929, 2004.

[HJW+08] T. Hachisuka, W. Jarosz, R. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. W. Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[JJN76] J. F. Jarvis, C. N. Judice, and W. H. Ninke. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Comp. Graph. and Image Proc.*, 5(1):13–40, 1976.

[KCODL06] J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski. Recursive Wang tiles for real-time blue noise. In *SIGGRAPH 2006 Proceedings*, pages 509–518, 2006.

[KEBS97] T.D. Kite, B.L. Evans, A.C. Bovik, and T.L. Sculley. Digital halftoning as 2-D delta-sigma modulation. In *IEEE Int. Conf. on Image Proc.*, pages 79–802, 1997.

[Kel97] A. Keller. Instant radiosity. In *SIGGRAPH '97 Proceedings*, pages 49–55, 1997.

[KK02] T. Kollig and A. Keller. Efficient multidimensional sampling. *Computer Graphics Forum*, 21(3):557–563, 2002.

[KK03] T. Kollig and A. Keller. Efficient illumination by high dynamic range images. In *EG Symp. on Rendering*, pages 45–51, 2003.

[LD06] A. Lagae and P. Dutré. An alternative for Wang tiles: colored edges versus colored corners. *ACM Trans. Graph.*, 25(4):1442–1459, 2006.

[LFCD07] Y.-C. Lai, S. Fan, S. Chenney, and C. Dyer. Photorealistic image rendering with Population Monte Carlo energy redistribution. In *EG Symp. on Rendering*, pages 287–296, 2007.

[Nie92] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, Pennsilvania, 1992.

[ODJ04] V. Ostromoukhov, C. Donohue, and P. M. Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. on Graph.*, 23(3):488–498, 2004.

[Ost07] V. Ostromoukhov. Sampling with polyominoes. In *ACM SIGGRAPH 2007 papers*, page 78, 2007.

[RCL+08] F. Rousselle, P. Clarberg, L. Leblanc, V. Ostromoukhov, and P. Poulin. Efficient product sampling using hierarchical thresholding. In *Computer Graphics International 2008*, pages 465–474, 2008.

[RFS03] J. Rigau, M. Feixas, and M. Sbert. Refinement criteria based on f-divergences. In *14th EG Workshop on Rendering*, pages 260–269, 2003.

[Shi91] P. Shirley. Discrepancy as a quality measure for sampling distributions. In *Eurographics '91*, pages 183–194, 1991.

[SKSP09] L. Szirmay-Kalos, L. Szécsi, and A. Penzov. Importance sampling with Floyd-Steinberg halftoning. In *Eurographics 09, Short papers*, pages 69–72, 2009.

[Stu81] P. Stucki. Mecca — a multiple-error correcting computation algorithm for bilevel hardcopy reproduction. Technical Report RZ1060, IBM Research Lab, Zurich, 1981.

[TCE05] J. Talbot, D. Cline, and P. K. Egbert. Importance resampling for global illumination. In *EG Symp. on Rendering*, pages 139–146, 2005.

[VG97] E. Veach and L. Guibas. Metropolis light transport. *SIGGRAPH '97 Proceedings*, pages 65–76, 1997.

[WFA+05] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: A scalable approach to illumination. In *SIGGRAPH 2005 Proceedings*, pages 1098–1107, 2005.

[Whi80] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

[WKB+02] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination using fast ray tracing. In *13th EG Workshop on Rendering*, pages 15–24, 2002.

[WWL05] L. Wan, T.T. Wong, and C.S. Leung. Spherical Q2-tree for sampling dynamic environment sequences. In *EG Symp. on Rendering*, pages 21–30, 2005.