EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel
(Guest Editors)

Volume 21 (*2002*), *Number 3*

A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm

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

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

Abstract

This paper presents a new mutation strategy for the Metropolis light transport algorithm, which works in the unit cube of pseudo-random numbers instead of mutating in the path space. This transformation makes the integrand have lower variation and thus increases the acceptance probability of the mutated samples. Higher acceptance ratio, in turn, reduces the correlation of the samples, which increases the speed of convergence. We use both local mutations that choose a new random sample in the neighborhood of the previous one, and global mutations that make "large steps", and find the samples independently. Local mutations smooth out the result, while global mutations guarantee the ergodicity of the process. Due to the fact that samples are generated independently in large steps, this method can also be considered as a combination of the Metropolis algorithm with a classical random walk. If we use multiple importance sampling for this combination, the combined method will be as good at bright regions as the Metropolis algorithm and at dark regions as random walks. The resulting scheme is robust, efficient, but most importantly, is easy to implement and to combine with an arbitrary random-walk algorithm.

1. Introduction

Global illumination algorithms have to identify those light paths that connect the light sources to the eye via reflections and refractions and integrate their image contribution for each pixel of the screen. Since the domain of such light paths has high dimension, Monte-Carlo quadrature rules are applied, which generate M random light paths $\mathbf{z}_1, \ldots, \mathbf{z}_M$ with probability density $p(\mathbf{z})$ and approximate the integrand as follows:

$$\Phi_j = \int_{\mathcal{P}} F(\mathbf{z})\, d\mathbf{z} \approx \frac{1}{M} \sum_{i=1}^{M} \frac{F(\mathbf{z}_i)}{p(\mathbf{z}_i)},$$

where Φ_j is the value stored in pixel j, $\mathcal{P}$ is the domain of the light paths, and $F(\mathbf{z})$ is the image contribution of path $\mathbf{z}$. Monte-Carlo quadratures obtain the result with certain variance. The variance can be reduced if the light paths are sampled with a probability density that is at least approximately proportional to the contribution of the light path. This variance reduction technique is called *importance sampling* [9]. Since a path carries light on several wavelengths, the contribution is a vector, thus the selection of a "proportional" probability density requires further considerations. In order to express where the elements of the vector are large, a *scalar contribution function* $I(\mathbf{z})$ is defined. This scalar contribution function can, for example, represent the luminance of the carried light. The goal is then to sample the domain of light paths with a probability density $p(\mathbf{z})$ that is proportional to the scalar contribution function: $p(\mathbf{z}) = I(\mathbf{z})/b$. Scalar b comes from the requirement of normalization: $b = \int_{\mathcal{P}} I(\mathbf{z})\, d\mathbf{z}$. Let us consider the relevant techniques to attack this problem, including local and multiple importance sampling and the Metropolis method.

1.1. Local and multiple importance sampling

Importance sampling should obtain light paths with a probability density that mimics the scalar contribution function. Since the contribution of the path is the product of the emission at the beginning and the cosine weighted BRDFs at the visited points, *local importance sampling* constructs the probability density step-by-step taking into account these factors independently. BRDF sampling[3] obtains a random direction that mimics the cosine weighted BRDF, light

source sampling[8] finds a point with a probability that is proportional to the emission, and Russian roulette generates the next step or obtains zero contribution with the probability of the reflection. In order to sample directions, points, termination, etc. of a particular path, we should take uniformly distributed pseudo or quasi random numbers in the unit interval and transform them to the path space. Those random numbers from which a complete path is generated can be considered as a point in a high-dimensional unit cube. Let us call this cube $\mathcal{U}$ as the *primary sample space*, and denote the transformation from here to the space of light paths by $\mathbf{z} = S(\mathbf{u})$. The pixel contribution can also be obtained as an integral in the primary sample space:

$$\Phi_j = \int_{\mathcal{U}} F(S(\mathbf{u})) \cdot \left| \frac{dS(\mathbf{u})}{d\mathbf{u}} \right| d\mathbf{u},$$

where

$$\left| \frac{dS(\mathbf{u})}{d\mathbf{u}} \right| = \frac{1}{p_S(\mathbf{u})}$$

is the Jacobi determinant of the inverse mapping. Intuitively, the Jacobi determinant expresses the local expansion between two corresponding spaces[2], thus if $\mathbf{u}$ is a uniformly distributed random variable, then $p_S(\mathbf{u})$ will be the probability density of $\mathbf{z} = S(\mathbf{u})$.

Unfortunately, p_S cannot be made exactly proportional to the scalar contribution function if the directions and the light source points are sampled independently. If the target of the next step is selected with light source sampling, then the probability density of the reflection direction will not be proportional to the cosine weighted BRDF. On the other hand, if the next ray is sampled with BRDF sampling, then the target point will not be distributed proportionally to the emission intensity. It means that although the weighted contribution $F(S(\mathbf{u}))/p_S(\mathbf{u})$ and the weighted scalar contribution function $I(S(\mathbf{u}))/p_S(\mathbf{u})$ have lower variation that F and I, respectively, but the weighted scalar contribution function is not constant.

Different importance sampling strategies decide differently what is sampled and what is computed from the sampled parameters, and thus might be better or worse to mimic certain types of light transfers. *Multiple importance sampling*[12] tries to get the best of these algorithms by combining several different sampling techniques in a way that the variance is further reduced. Suppose that we can sample a light path $\mathbf{z}$ with N different sampling techniques associated with $p_1(\mathbf{z}), \ldots, p_N(\mathbf{z})$ densities and $M_1, \ldots, M_N$ sample numbers. If these methods were used alone, then method n would weight integrand sample $F(\mathbf{z})$ with $1/(p_n(\mathbf{z})M_n)$. If this sampling technique is bad for a given sample $\mathbf{z}'$, then it can happen that $p_n(\mathbf{z}')$ is small while $F(\mathbf{z}')$ is not small, which would result in a few but very large contributions $F(\mathbf{z}')/(p_n(\mathbf{z}')M_n)$ and consequently high variance. The combination strategy of multiple importance sampling emphasizes a given method where it is good, thus eliminates these bad samples. *Balance heuristic* would weight the integrand by the average probability of all sampling techniques, i.e. by $1/(\sum_n p_n(\mathbf{z})M_n)$. *Maximum heuristic*, on the other hand, would consider $F(\mathbf{z}')/(p_n(\mathbf{z}')M_n)$ only if $p_n(\mathbf{z}')M_n$ is the maximum for different n values. Thus if at least a single elementary sampling technique is good for a particular sample $\mathbf{z}'$ (i.e. its density is not small if the contribution is large), then the weighted average will contain many, not too large samples, which reduces the variance. Thus multiple importance sampling can further reduce the variation of the weighted contribution, but it is still far from being constant.

1.2. Metropolis light transport

Unlike local importance sampling, Metropolis method can sample a complete path as a whole, not just the steps of this path[14]. It requires no a-priori knowledge to construct a probability density function in advance, but converges to the required probability density automatically.

Metropolis method constructs a Markovian process whose stationary distribution is $p(\mathbf{z})$ to generate samples according to $p(\mathbf{z}) = 1/b \cdot I(\mathbf{z})$. The next state $\mathbf{z}_{i+1}$ of this process is found by letting an almost arbitrary *tentative transition function* $T(\mathbf{z}_i \to \mathbf{z}_t)$ generate a *tentative sample* $\mathbf{z}_t$ which is either accepted as the real next state or rejected making the next state equal to the actual state. The decision uses the "*acceptance probability*" $a(\mathbf{z}_i \to \mathbf{z}_t)$ that expresses the increase of the scalar contribution function (if this "acceptance probability" is greater than 1, then the sample is accepted deterministically). The formal definition of this Markovian process $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_i, \ldots\}$ is as follows:

for $i = 1$ **to** M **do**
  Based on $\mathbf{z}_i$, sample a tentative point $\mathbf{z}_t$ using $T(\mathbf{z}_i \to \mathbf{z}_t)$
  $a(\mathbf{z}_i \to \mathbf{z}_t) = \frac{I(\mathbf{z}_t) \cdot T(\mathbf{z}_t \to \mathbf{z}_i)}{I(\mathbf{z}_i) \cdot T(\mathbf{z}_i \to \mathbf{z}_t)}$
  // *accept with probability* $a(\mathbf{z}_i \to \mathbf{z}_t)$
  Generate random number r in $[0,1]$.
  if $r < a(\mathbf{z}_i \to \mathbf{z}_t)$ **then** $\mathbf{z}_{i+1} = \mathbf{z}_t$
  else $\mathbf{z}_{i+1} = \mathbf{z}_i$
endfor

Veach and Guibas[14] recognized that the basic Metropolis algorithm needs to be modified to make it suitable for the solution of the global illumination problem. Such modifications allowed a single process to be used simultaneously for all pixels, the reduction of the start-up bias and the utilization of the rejected samples. Let us examine these modifications separately.

Global illumination requires a separate integral to be solved for each pixel. For pixel j, integrand $F(\mathbf{z})$ is the product of the measurement function of this pixel and the radiance carried by this path. In order to use a single process for all pixels, function I can be defined as the scalar contribution to any pixel of the screen, that is, I mimics only the radiance and will be independent of the pixel measuring function. In

this way, the number of samples contributing to a single pixel will be proportional to the luminance of this pixel.

Metropolis sampling converges to the desired probability distribution, but at the beginning of the process the samples are not selected with the required probability, which introduces some error in the estimation. This error is called as the *start-up bias*[11]. The original Metropolis light transport algorithm uses the following solution for the problem. In a preprocessing phase random samples are generated and the initial seed of the Metropolis algorithm is selected from this random population with a probability that is proportional to the scalar contribution function. Since in this case even the first sample follows the desired distribution, the start-up bias problem is eliminated in a statistical sense, i.e. it is converted to noise.

Veach recognized that it is worth using also the rejected samples since they also provide illumination information. Note that a tentative sample is accepted with probability a, while the original sample is kept with probability $1-a$. Replacing this random variable by its mean, both locations can be contributed but the contributions of the tentative sample and the old sample should be weighted with a and $1-a$, respectively.

Summarizing, the pseudo-code of the Metropolis light transport algorithm is as follows:

Generate path seeds
Approximate $b = \int I(\mathbf{z})\, d\mathbf{z}$ from the seeds
Find $\mathbf{z}_1$ from the seeds using $I(\mathbf{z})$
for $i = 1$ **to** M **do**
  Based on $\mathbf{z}_i$, sample a tentative point $\mathbf{z}_t$ using $T(\mathbf{z}_i \to \mathbf{z}_t)$
  $a(\mathbf{z}_i \to \mathbf{z}_t) = \min\left\{\frac{I(\mathbf{z}_t)\cdot T(\mathbf{z}_t \to \mathbf{z}_i)}{I(\mathbf{z}_i)\cdot T(\mathbf{z}_i \to \mathbf{z}_t)}, 1\right\}$
  Select pixel j to which $\mathbf{z}_i$ contributes
  Φ_j += $\frac{b}{M} \cdot \frac{F(\mathbf{z}_i)}{I(\mathbf{z}_i)} \cdot (1 - a(\mathbf{z}_i \to \mathbf{z}_t))$
  Select pixel k to which $\mathbf{z}_t$ contributes
  Φ_k += $\frac{b}{M} \cdot \frac{F(\mathbf{z}_t)}{I(\mathbf{z}_t)} \cdot a(\mathbf{z}_i \to \mathbf{z}_t)$
    // accept with probability $a(\mathbf{z}_i \to \mathbf{z}_t)$
  Generate random number r in $[0,1]$.
  if $r < a(\mathbf{z}_i \to \mathbf{z}_t)$ **then** $\mathbf{z}_{i+1} = \mathbf{z}_t$
  else $\mathbf{z}_{i+1} = \mathbf{z}_i$
endfor

Unlike other Monte-Carlo algorithms, the Metropolis algorithm generates not statistically independent, but correlated samples. The statistical independent sampling of Monte-Carlo quadrature guarantees that if the standard deviation of random variable $F(\mathbf{z})/p(\mathbf{z})$ is $\sigma_{primary}$, then the standard deviation of the Monte-Carlo quadrature will be $\sigma_{primary}/\sqrt{M}$ after evaluating M samples (the standard deviation is a good measure of the integration error). In case of correlated samples, the standard deviation of the quadrature can be bounded according to the Bernstein theorem[7]:

$$\sigma \leq \sigma_{primary} \cdot \sqrt{\frac{1 + 2\sum_{k=1}^{M} R(k)}{M}}$$

where $R(k)$ is an upper-bound of the correlation between $F(\mathbf{z}_i)/p(\mathbf{z}_i)$ and $F(\mathbf{z}_{i+k})/p(\mathbf{z}_{i+k})$. This formula shows that the correlation of the samples can increase the error [11, 1].

Let us consider what it means from the point of view of mutation strategies. If the mutations are small, then the next sample has no chance to be relatively independent of the previous one, thus the correlation will be high. Interestingly, large mutations can also lead to highly correlated samples (Figure 1). Suppose that the process has found a peak of the integrand. Having made a large perturbation, the scalar contribution function of the tentative sample will be much lower, thus the chance of accepting it will also be low. The point on the peak remains to be the sample point for many steps, which is responsible for high correlation.

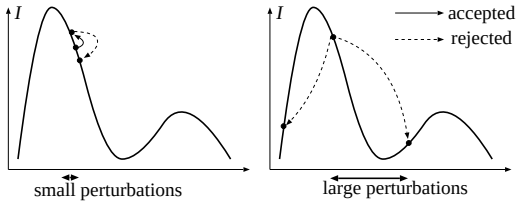


Figure 1: *Both large and small mutations can result in high correlation.*

Considering this, we can conclude that a single, constant mutation strategy cannot provide an effective algorithm. Instead we need a mutation strategy that is generally large but gets smaller around the peaks of the integrand. The original Metropolis light transport algorithm proposes the random combination of several strategies, each of them being tailored for a particular type of light transfer [14]. The construction and the combination of these elementary strategies requires care and usually involves scene dependent parameter tuning. In this paper we present a simpler method, which automatically results in efficient mutations.

2. Finding a good space for making mutations

The conclusions of the previous section immediately lead to the following question. Is it possible to adapt the mutation strategy itself such that the perturbation gets automatically smaller when we are around the peak and thus anticipate that larger perturbations will be rejected? At the first glance, the answer is negative since to construct such tentative transition probabilities, explicit knowledge of the scalar contribution function would be needed, which is not available. However, if the domain and consequently the integrand is transformed in a way that the integrand has lower variation, the number of rejections can greatly be reduced (note that the rejection probability is proportional to the ratio of contributions). Such transformation would obviously expand the domain where the original integrand is large and shrink it

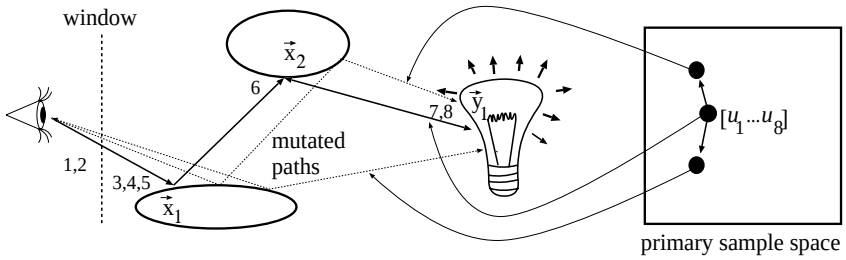


Figure 2: *The correspondence between the mutations in the primary sample space and in the path space.*

where the original integrand is small. If constant size perturbations are used in the transformed domain, they would correspond to larger steps where the original integrand is large and smaller steps where it is small.

Such transformation is for free if the random paths are built using importance sampling, e.g. BRDF sampling, light source sampling, Russian roulette and multiple importance sampling. As concluded, these techniques transform the points of the *primary sample space* in order to place samples in path regions approximately proportional to their contribution. This transformation makes both the new integrand

$$F^*(\mathbf{u}) = F(S(\mathbf{u})) \cdot \left| \frac{dS(\mathbf{u})}{d\mathbf{u}} \right| = \frac{F(S(\mathbf{u}))}{p_S(\mathbf{u})}$$

and the new scalar contribution function

$$I^*(\mathbf{u}) = \frac{I(S(\mathbf{u}))}{p_S(\mathbf{u})}$$

have lower variation, i.e. relatively flat. This is exactly what we need. Thus the perturbation strategy that works in the primary sample space will adapt to the properties of the integral and reduces the perturbation size where the integral is large.

Let us take an example to show how the perturbation in the primary sample space affects the path. The example can also be followed in Figure 2. For the sake of simplicity, assume that we use path tracing with direct light source computation at the end of the path in order to find a complete light path (the proposed algorithm can work with other random walk algorithms as well). Using two pseudo-random numbers u_1, u_2 a random point on the window is found and a ray is traced through this point, which finds surface point $\vec{x}_1$. At $\vec{x}_1$ we take u_3 to randomly select a BRDF from the elementary BRDFs composing the reflection function at $\vec{x}_1$ (e.g. diffuse + specular) and to decide whether or not the walk has to be terminated according to Russian roulette. Assume that we decided to continue the walk, thus we use another two random numbers u_4, u_5 to sample the direction with a density that is approximately proportional to the selected elementary BRDF. The obtained direction with starting point $\vec{x}_1$ define a new ray that is traced to find the new point $\vec{x}_2$. Here we decide again on the termination using u_6. Suppose that the random number u_6 and the albedo at $\vec{x}_2$ are such that the walk is terminated. In order to apply direct light source estimation, a point $\vec{y}_1$ is sampled on the surface of the light source using random numbers u_7, u_8. This light source point is connected with the last point $\vec{x}_2$ with a shadow ray that tests if the light source point is visible from here. If the two points are not occluded, we have established the following light path: $\mathbf{z} = (\vec{y}_1, \vec{x}_2, \vec{x}_1, \text{e}\vec{\text{y}}\text{e})$. Clearly, this path is unambiguously defined by the vector $\mathbf{u} = (u_1, \ldots, u_8)$. The mapping $\mathbf{z} = S(\mathbf{u})$ is defined by the path ray-tracing scheme that involves BRDF sampling, Russian roulette and light source sampling.

Let us now slightly perturb the elements of primary sample $\mathbf{u} = (u_1, \ldots, u_8)$. Decision parameters u_3 and u_6 are used to terminate the walk and to select from elementary BRDFs. If the perturbations of these values fit in the range allowed by the albedos and the perturbation of positions and directions do not alter the respective visibilities, then the new values lead to the same decisions, thus the structure of the path is not altered (for example, the new path will also connect 4 points). However, when one of the new values steps over the albedo limits or alter the visibility relationships, then the remaining part of the sub-path is cut or the previously terminated walk has to be continued.

Since path tracing with direct light source computation at the end is far from being the most efficient strategy to sample paths, we also implemented the proposed idea with bidirectional path tracing[12]. This method produces an eye path and a light path and connects all points of the two paths with each other with deterministic shadow rays (Figure 3). This algorithm produces many paths and each of them can also be generated differently by the same algorithm. For example, the path connecting the eye via $\vec{x}_1$ to light source point $\vec{y}_1$ contains a random ray between the eye and $\vec{x}_1$ and a deterministic shadow ray between $\vec{x}_1$ and $\vec{y}_1$. This path could also be obtained with a different probability in a way that the deterministic shadow ray is between the eye and $\vec{x}_1$ and the connection of $\vec{x}_1$ and $\vec{y}_1$ is found randomly. In order to get the best from these sampling techniques, we can use the maximum heuristic of multiple importance sampling. The reason of our preference of maximum heuristic over balanced heuristic here is that maximum heuristic does not require all deterministic shadow rays to be traced. Maximum

heuristic decides whether or not the weight of the sample is non zero according to the path probabilities that can be computed from the local BRDFs and from the light source selection scheme, and are independent of the occlusions of the shadow rays. The visibility is checked only if the weight turns out to be positive and thus there is a chance for the contribution to be not zero.

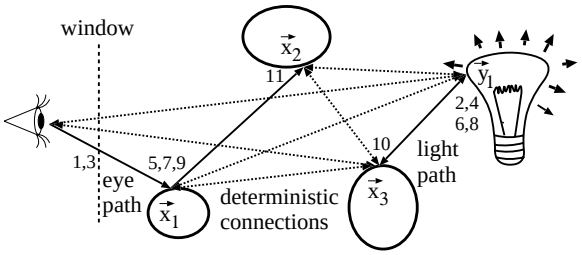


Figure 3: *A sample path generated by bi-directional path tracing.*

Bi-directional path tracing produces many path from a single primary sample (for example, in Figure 3 the eye path connects three points and the light path connects two points, which can be combined in six different ways). Thus the definition of the scalar contribution function for such family of paths requires additional considerations. One alternative would be the sum of the luminances of the elementary paths. However, this would prefer long paths since longer eye and light paths allow more combinations to be made, which would increase the scalar contribution function and consequently the selection probability. Working with long and complex paths is not efficient computationally, thus a better alternative for the scalar contribution function is the maximum of the luminances of the elementary paths.

The proposed method is efficient if a small mutation in the primary sample space corresponds to a small mutation in the path space. However, when Russian roulette changes the length of the eye path, then the coordinates of the eye path might be assigned to the light path or vice versa, which can cause large changes in the path space. To solve this problem, the coordinates assigned to eye and light paths should be separated. For example, coordinates of odd and even indices can be used separately to define the eye path and the light path, respectively (Figure 3).

3. Large steps

The next problem that needs special care comes from the regions of zero contribution. The Markovian process used in Metropolis algorithm should be ergodic, i.e. all samples of non-zero contribution should be generated sooner or later with positive probability. In the global illumination setting it is very likely that light paths of non-zero contribution form islands in the path space. If the mutations are not big enough to jump from one island to the other, then the ergodicity condition cannot be met. In order to solve this problem, we include completely independent steps in the algorithm, which obtain a tentative sample without considering what the actual state is. These global mutations are called *large steps*. Large steps have three different merits. They generate any non-zero contribution point with positive probability, thus the requirement of ergodicity is met. If a large step is accepted, then Metropolis process starts from a new random seed, which significantly reduces the start-up bias error. Finally, the probability density of the tentative samples obtained with large steps is known, which allows for their sophisticated secondary utilization according to the concept of multiple importance sampling.

Let us consider how the tentative samples are worth generating in these large steps. In order to reduce correlation, the acceptance probability should be maximized. Note that the tentative probability of large steps depends only on the target state, that is $T(\mathbf{z}_i \to \mathbf{z}_t) = T(\mathbf{z}_t)$. Considering this, the acceptance probability is:

$$a(\mathbf{z}_i \to \mathbf{z}_t) = \frac{I(\mathbf{z}_t) \cdot T(\mathbf{z}_i)}{I(\mathbf{z}_i) \cdot T(\mathbf{z}_t)}.$$

This probability could be set to 1 if $T(\mathbf{z})$ were proportional to the scalar contribution function, but this would require the explicit knowledge of the scalar contribution function. If the large steps are examined in the primary sample space, then the acceptance probability has the following form:

$$a(\mathbf{u}_i \to \mathbf{u}_t) = \frac{I^*(\mathbf{u}_t) \cdot T^*(\mathbf{u}_i)}{I^*(\mathbf{u}_i) \cdot T^*(\mathbf{u}_t)}.$$

Due to importance sampling $I^*(\mathbf{u}_t)$ is usually quite flat. More precisely, it is as flat as BRDF sampling, light source sampling and Russian roulette can describe the importance of a path. Using the — rather crude — approximation that the transformed scalar contribution function is constant, we obtain:

$$a(\mathbf{u}_i \to \mathbf{u}_t) \approx \frac{T^*(\mathbf{u}_i)}{T^*(\mathbf{u}_t)}.$$

Thus the acceptance probability will be close to 1 if the mutation function of the large steps are realized with a uniform probability.

4. Designing mutation strategies in the infinite dimensional cube: lazy evaluation

So far, we neglected the fact that the primary sample space, i.e. the cube where the sample points are perturbed, is infinite dimensional, where it is impossible to unambiguously define a point with finite numbers. However, this is not a problem if the coordinates of each point are evaluated in a lazy way. Note that a point unambiguously defines a light path according to the particular random walk algorithm incorporating

BRDF sampling, Russian roulette and light source sampling, but only the first few coordinates are used until the paths are terminated according to Russian roulette. Let us evaluate, store and perturb only those coordinates of the current point, which have been needed by the longest path happened so far. If it turns out that the current path is longer than the longest path encountered so far and thus we need new coordinates, then we have to repeat the life history just for these new coordinates. We have to go back to their last use or if they have not been used before, to the last accepted large step since the last step generated all coordinates randomly and independently of the former states (in this sense accepted large steps are those critical time instances beyond which we do not have to remember). If a coordinate has been never used, then its initial value is obtained randomly as the large step would have obtained it. Then the history happened since the last use is played again, i.e. the coordinate is perturbed by the times of the accepted perturbations happened since the last use. Note that only the number of accepted mutations counts since rejected mutations do not affect the future.

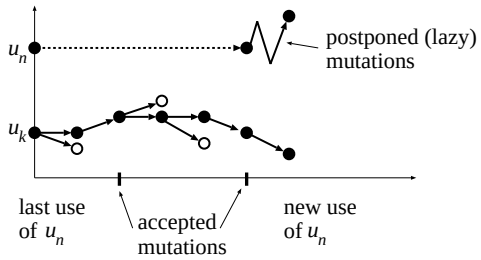


Figure 4: *Lazy evaluation of the coordinates: the mutations of the not used coordinates are postponed until they are needed again.*

Let us define a counter called *time* for the global time of the process which counts the number of accepted mutations. Each coordinate is associated with a time-stamp called *modify* that stores the global time when this coordinate was modified most recently. The time of the last accepted large step is stored in variable *large_step_time*. When a new coordinate is needed, it is checked whether this coordinate has been used before. If not, it is initialized as a random number and its time stamp is set to *large_step_time*. If it has already been used, the value of the coordinate was set at time *modify*. Then the coordinate is perturbed by *time* − *modify* times (Figure 4). The implementation of this algorithm is presented in Section 6.

5. Utilization of rejected and large step samples

The original Metropolis light transport algorithm makes use of rejected samples by replacing the random variable of the random acceptance by its mean, and multiplies the original and tentative values by the rejection and acceptance probabilities. However, large steps allow us to do it even better according to multiple importance sampling. Suppose that we use two sampling techniques, and the first is the Metropolis method that generates M_1 number of samples with probability $p_1(\mathbf{u})$, while the second is the uncorrelated sampling of large steps that compute M_2 samples with probability $p_2(\mathbf{u})$. When they are combined, balance heuristic[13] would weight the integrand samples by the average selection density, that is by

$$w(\mathbf{u}) = \frac{1}{M_1 p_1(\mathbf{u}) + M_2 p_2(\mathbf{u})}.$$

The combination requires the knowledge of the sampling probabilities of both methods. Metropolis algorithm generates a sample with the probability that is proportional to the scalar contribution function:

$$p_1(\mathbf{u}) = \frac{I^*(\mathbf{u})}{b}.$$

For large steps, uniform point selection is used, thus:

$$p_2(\mathbf{u}) = 1.$$

If the probability of large steps is p_{large}, then the number of samples of the second method is expected to be p_{large} times the number of samples of the first method, thus $M_1 = M$ and $M_2 = p_{large}M$.

Finally we should take into account that due to mean value substitution the new, tentative samples of Metropolis method will be weighted by acceptance probability a, while the old samples are also used with weight $1-a$. Putting these altogether, the contribution of an old Metropolis sample is

$$(1-a)\cdot w(\mathbf{u})\cdot F^*(\mathbf{u}) = \frac{(1-a)\cdot F^*(\mathbf{u})}{(I^*(\mathbf{u})/b + p_{large})M}.$$

When a large step is made, the generated path is a sample of the Metropolis method and a sample of the uncorrelated strategy at the same time. The Metropolis sample should be weighted by a according to the mean value substitution, thus the combined contribution of the two techniques is

$$a\cdot w(\mathbf{u})\cdot F^*(\mathbf{u}) + w(\mathbf{u})\cdot F^*(\mathbf{u}) = \frac{(a+1)\cdot F^*(\mathbf{u})}{(I^*(\mathbf{u})/b + p_{large})M}. \quad (1)$$

Small steps can stem only from the Metropolis method, thus their contribution is

$$a\cdot w(\mathbf{u})\cdot F^*(\mathbf{u}) = \frac{a\cdot F^*(\mathbf{u})}{(I^*(\mathbf{u})/b + p_{large})M}. \quad (2)$$

Equations (1) and (2) can be merged together if we suppose that variable $large\_step$ is 1 if a large step is made and zero otherwise, thus the contribution of a new, i.e. tentative sample is

$$\frac{(a + large\_step)\cdot F^*(\mathbf{u})}{(I^*(\mathbf{u})/b + p_{large})M}.$$

Note that Metropolis method is good in generating bright image sections while poor at dark regions since the number of samples in a pixel is proportional to the luminance of this

pixel. Random walks, on the other hand, provide pixel values with approximately the same relative error if they use the same number of samples in each pixel. Thus when relative or other perceptual error metric is used, random walks are better for darker regions. The proposed combination can thus improve dark image areas. We expect that the combined method is as good as Metropolis sampling in bright sections, but can also handle dark regions as well as random walks.

6. Implementation details

Suppose that we already have a random walk implementation. In order to build the proposed Metropolis sampler into this algorithm, the pseudo or quasi random number generator should be replaced, and the weighting of the light path contributions should be altered.

Let us first consider the primary sample generator that obtains a number in the unit interval in order to integrate according to the *i*th coordinate of the integrand. Function `PrimarySample` takes coordinate index `i` and also uses a global variable called `large_step` that is 1 if the actual mutation is a large step and 0 otherwise, and obtains the actual value of coordinate `i`. The algorithm implements the lazy evaluation method and pushes the previous values onto a stack of the index–coordinate pairs, which should be restored later if this sample is rejected by the Metropolis method. The variable names are according to section 4, vector `u` is the sample in the primary sample space, and each of its coordinates has a `value` in [0,1] and a local time stamp called `modify`.

```
float PrimarySample(int i) {
  if (u[i].modify < time) {
    if (large_step) {         // large step
      Push(i, u[i]);          // save state
      u[i].modify = time;
      u[i].value = random();
    } else {                  // small step
       if (u[i].modify < large_step_time) {
         u[i].modify = large_step_time;
         u[i].value = random();
       }
            // lazy evaluation of mutations
       while (u[i].modify < time-1) {
         u[i].value = Mutate(u[i].value);
         u[i].modify++;
       }
       Push(i, u[i]);         // save state
       u[i].value = Mutate(u[i].value);
       u[i].modify++;
    }
  }
  return u[i].value;
}
```

This subroutine calls the `Mutate` function to find a mutated coordinate. As proposed in [14], the mutations are controlled by the $s = s_2 \cdot \exp(-\log(s_2/s_1)U)$ formula, where U is a uniformly distributed random variable in [0,1] and the samples are expected in $[s_1, s_2]$. The following mutation function gets the actual value and returns the mutated one:

```
float Mutate( float value ) {
  float s1 = 1./1024, s2 = 1./64;
  float dv = s2*exp(-log(s2/s1)*random());
  if (random() < 0.5) {
    value += dv; if (value > 1) value -= 1;
  } else {
    value -= dv; if (value < 0) value += 1;
  }
  return value;
}
```

In order to compute the contribution of the generated light paths, the mean value substitution and the proposed multiple importance sampling technique have been implemented. Recall that in these cases each step results in two samples that should be contributed. However, an accepted sample should be contributed at least two times, when it is born and when it gets old. Thus we can still work with the contribution of a single path at a time if the weight `w` of the multiple contributed samples are cumulated. The following `Next` function handles both rejected and accepted samples, but returns only one of them to be contributed to the affected pixel. If rejection happens, the returned sample is the rejected one, since it will be invalid in the next cycle. However, if the sample is accepted, then the contributed sample is the old sample while the weight of the new sample will be increased in the next cycle. The function gets the affected pixel and its contribution $F^*(\mathbf{u})$ in variable `contrib`, and the transformed scalar contribution $I^*(\mathbf{u})$ in variable `I`:

```
Sample Next(float I, Contrib contrib) {
  float a = min(1, I/oldI);  // accept prob.
  newsample.contrib = contrib;
  newsample.w = (a+large_step)/(I/b+plarge)/M;
                          // cumulate weight
  oldsample.w += (1-a)/(oldI/b+plarge)/M;

  if (random() < a) {            // accept
    oldI = I;
    contribsample = oldsample;
    oldsample = newsample;
    if (large_step) large_step_time = time;
    time++;
    ClearStack();  // no state restoration
  } else {                       // reject
    contribsample = newsample;
                  // restore state
    while( !IsStackEmpty() ) Pop(i, u[i]);
  }
  large_step = (random() < plarge) ? 1 : 0;
  return contribsample;
}
```

If the path contributes to more than one pixels (as can happen in bi-directional path tracing), then variable `contrib` is an array. Note that we could remove the tentative transition probabilities from the formula of the acceptance probability since our small and large steps apply symmetric tenta-

tive transition functions. When the mutation is accepted, the stack of index–coordinate pairs is cleared by `ClearStack` without restoring its values. Upon rejection, on the other hand, the algorithm returns to the original sample by popping the index–coordinate pairs of the point in the primary sample space.

The color of the pixels are updated according to the returned sample of this function. The affected pixels and their contributions can be read from `contrib`, then the contributions are multiplied with their weights `contrib.w` and added to the respective pixels.

7. Evaluation of the new method

In order to evaluate the proposed method, we implemented it with path tracing, with simple bi-directional path tracing, which was used in the original Metropolis light transport algorithm, and also with normal bi-directional path tracing[4, 12]. The main difference of the simple and normal bi-directional path tracing algorithms is that the simple version connects the end points of the shooting and gathering paths, while the normal version establishes connections between all visited points of the gathering and shooting paths, respectively.

First the standard Cornell Box scene was used for the error analysis. In order to make the lighting difficult, the back wall is a glossy mirror (the exponent of the Phong BRDF is 50) and the specularity of the surfaces is high.

The original Metropolis light transport applied mutations that replaced either the complete path or only a sub-part of the path while the rest part remained unchanged. On the one hand, such partial path mutations require less number of rays to be shot than those mutations which would modify all rays composing the path. On the other hand, these limited mutations increase the correlation and explore just "planes" in the high-dimensional integration domain. Thus if the mutations change the path only partially, then we need more but cheaper samples to obtain the solution. Since the proposed method mutates all coordinates simultaneously and generates many paths from a single primary sample if normal bi-directional path tracing is used, the new methods work with more expensive samples. This means that in order to get a fair comparison, we have to plot the error functions against time and not against the number of mutations.

In order to evaluate the efficiency of mutating in the primary sample space and multiple importance sampling, we took three different algorithms. The first one perturbed the samples in the path space, and both the original and the tentative samples were used in the quadrature according to mean value substitution, which corresponds to the original Metropolis light transport algorithm. In the second case, we perturbed the samples in the primary sample space and allowed large steps but still used the original mean value substitution for old and tentative samples. In the third experiment the large steps were taken into consideration with multiple importance sampling. First we allowed large mutations, i.e. a single mutation could cover about 20 percent of the domain of the corresponding variable. As expected, mutating in the primary sample space significantly increased the acceptance probability. The acceptance ratios of the original and the new algorithms were 43% and 76%, respectively. This higher acceptance ratio reduces the start-up bias and also the variance of the algorithm. However, when the perturbation size is small, the gain received from the transformation becomes smaller. For example, the average acceptance probability increased from 62% to 84% when the interval of perturbation was only one percent of the allowed domain. Finally, the perturbation size was set to 5% and the errors of the algorithms were measures as functions of time. The results are shown in the left of Figure 5.

Note that mutating in the primary sample space improved the convergence but the benefit of multiple importance sampling is not obvious yet. The reason is that we used the RMS error, which is not really good at expressing image differences. For example, if a dark pixel has some σ^2 variance, then it would mean the same RMS error as if a bright pixel had it. However, this variance is much more noticeable in the dark pixels. To overcome this problem, a perceptual error measure was also used [6]. For instance, according to Weber law, the human eye is sensitive to relative rather than absolute errors. Thus we computed the relative error in all pixels and defined the perceptual error as the ratio of those pixels where the relative error exceeded a given threshold. The right of Figure 5 has been plotted with this metric, and shows the superior performance of multiple importance sampling.

7.1. Testing on complex scenes

The proposed method has also been tested on complex scenes, for example, on the Buddha scene (Figure 8 in the color section) that consists of nearly 300 thousand triangles. The lighting is especially difficult at the back of the Buddha since it can receive illumination through a reflection of the mirror, the Buddha itself is highly specular (the RMS slope in the Cook-Torrance BRDF is 0.05) and we can see the mirror reflection of the back of the Buddha (these light paths contain two ideal and one highly specular reflections). The curved mirror consists of planar parts.

In order to render this scene, the normal bi-directional path tracing was equipped with the proposed Metropolis sampler. The average computation time of a single mutation took 0.2 msec on a Pentium III/1Ghz computer. Figure 8 of the color section has been rendered with only 25 mutations per pixel on average. Note that when the large step probability is small, the high correlation of the subsequent light paths results in "waves" in the image. Increasing the large step probability the waves disappear, but for very high large step probability, the dot noise becomes more significant at bright regions. This is due to the fact that large steps correspond to global mutations, thus the process is unable to

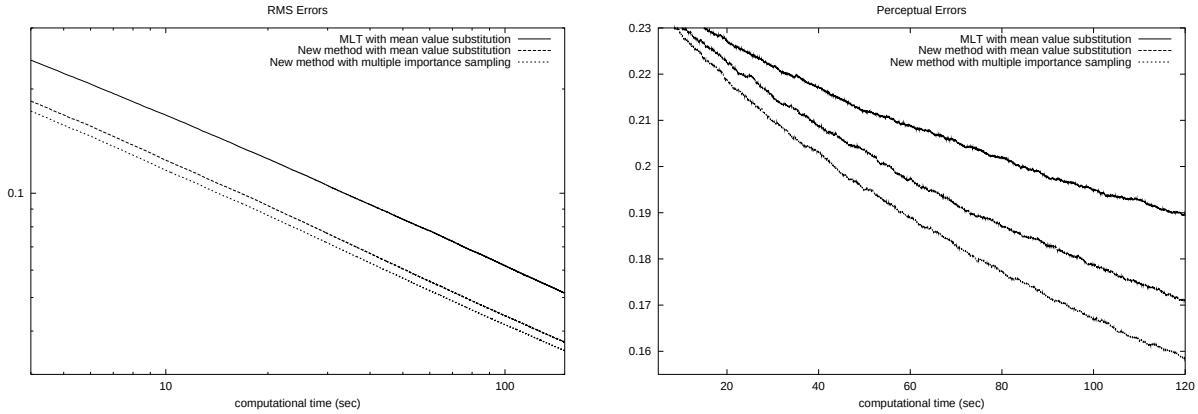


Figure 5: *Error curves of the original Metropolis light transport and the new Metropolis sampler included in simple bi-directional path tracing, using RMS measure (left), and a perceptual error measure (right).*

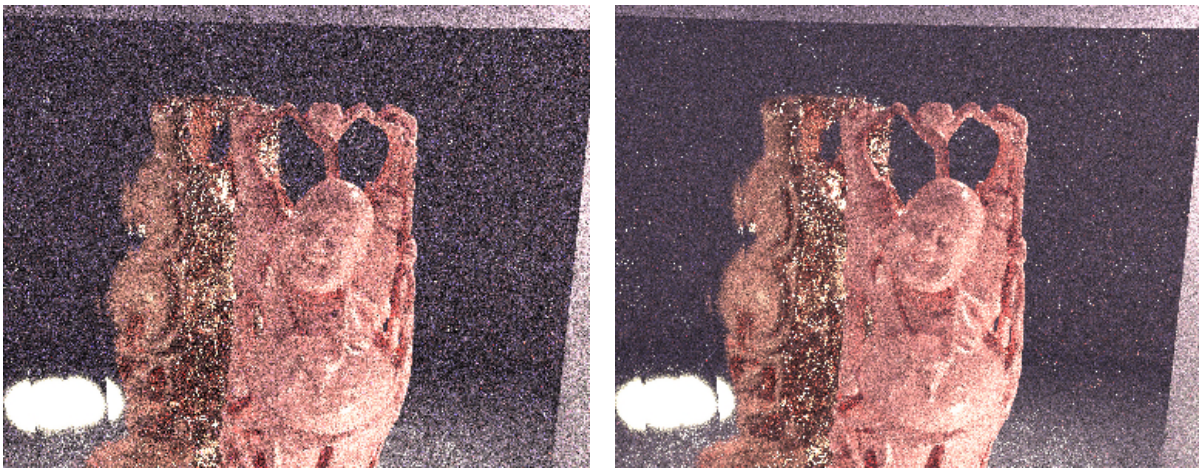


Figure 6: *Images rendered without (left) and with (right) multiple importance sampling using 25 mutations per pixel. The large step probability is 0.5.*

explore the space with fine, small steps. The optimal large step probability is about 0.5.

Figure 6 shows the effect of combining large steps with multiple importance sampling, which is really significant if the image contains high luminance variations (note the large difference between the reflected light source and the rest of the mirrorring wall behind the Buddha). The difference of bright sections of the two images is negligible, but darker pixels can greatly benefit from multiple importance sampling.

The proposed method has also been compared with pure bi-directional path tracing. Figure 9 of the color section shows the same scene rendered with 100 average number of samples by bi-directional path tracing and the new method, respectively. The new method provides better results using the same computational time, especially at difficult parts of the image.

Figure 7 shows a scene inspired by “disco lights”. The sphere-like object consists of 80 ideally reflecting and refracting triangles that are lit by 8 color light sources. Note that 256 mutations per pixel were enough to render the scene quite accurately.

8. Conclusions

This paper presented a new mutation method for the Metropolis light transport algorithm. The mutations are

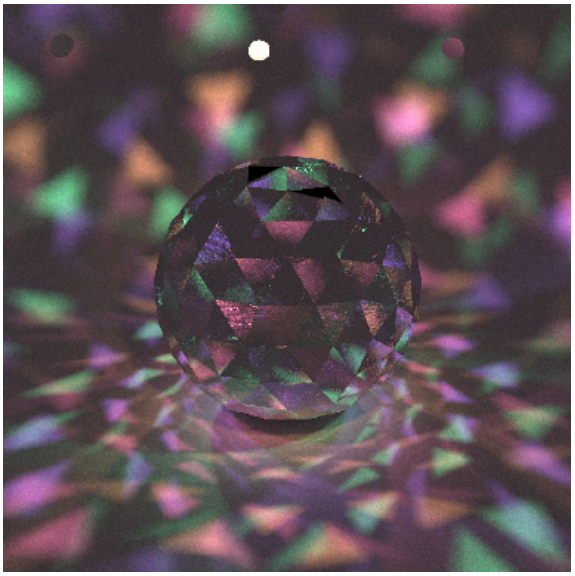


Figure 7: *Disco lights scene rendered with 256 mutations per pixel.*

computed in the primary sample space where other Monte-Carlo methods obtain the pseudo-random numbers. If importance sampling is used, then this strategy makes the integrand flatter, and increases the average acceptance probability and thus reduces the correlation. The other main advantage of this approach is that it does not require sophisticated techniques and tricks to set parameters. We also proposed the application of large steps to include independent samples in the sequence. These large, independent steps have threefold benefits. They reduce the start-up bias, guarantee the ergodicity and allow a more sophisticated reuse of tentative samples in the integral quadrature, which is based on the concepts of multiple importance sampling.

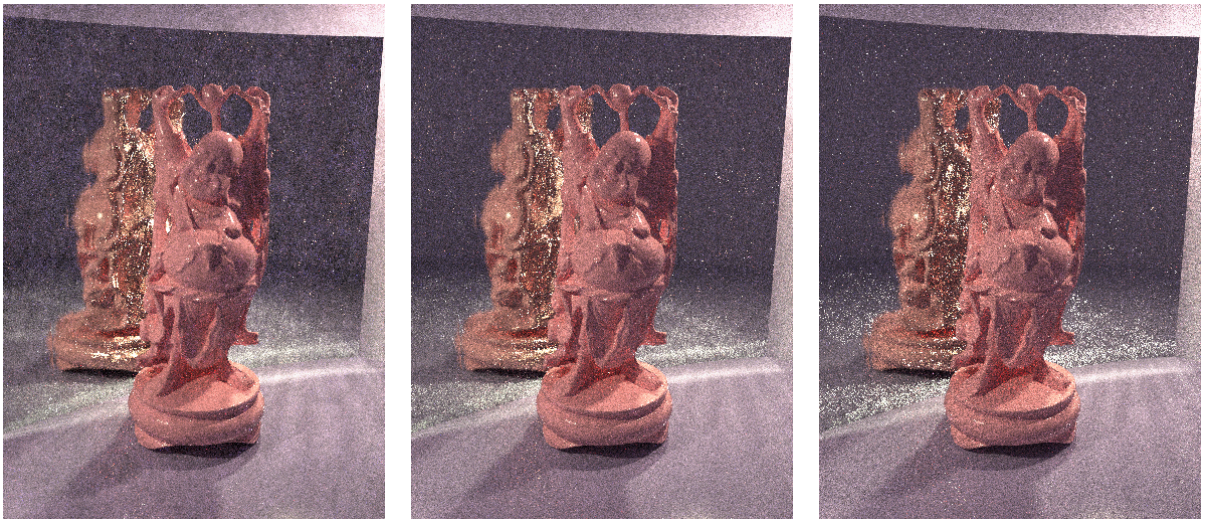
In our approach we made a clear distinction between the generation of primary samples according to the Metropolis method and the construction of light paths from them. In this way, any random walk algorithm can be equipped with the proposed general Metropolis sampler, just the random number generation and the scheme of weighting the light path contributions should be modified.

9. Acknowledgements

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

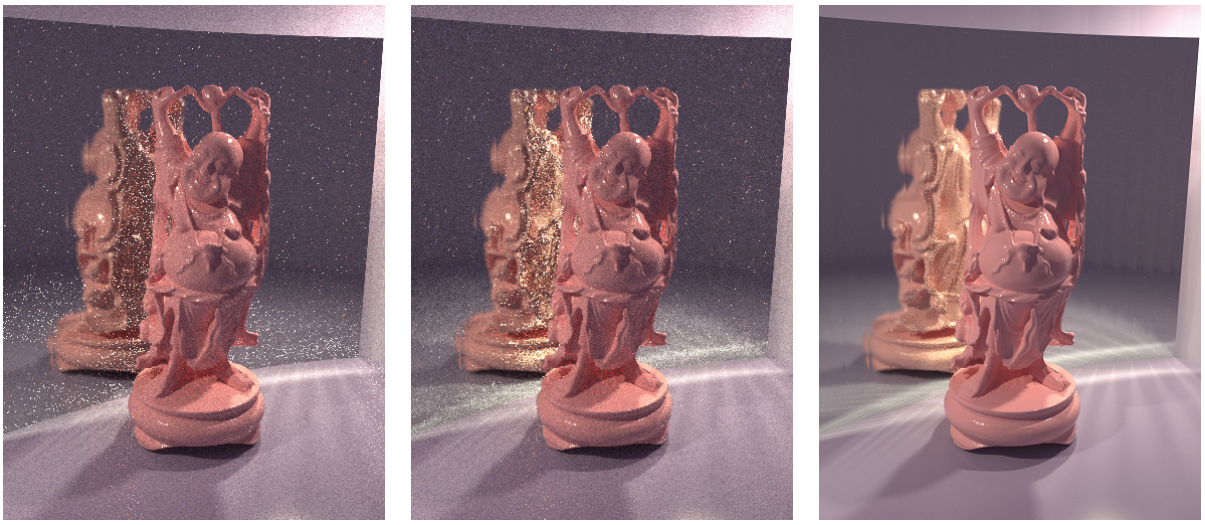
References

1. M. Ashikhmin, S. Premoze, P. Shirley, and B. Smits. A variance analysis of the Metropolis light transport algorithm. *Computers & Graphics*, 25(2):287–294, 2001. 3
2. I. Deák. *Random Number Generators and Simulation*. Akadémia Kiadó, Budapest, 1989. 2
3. J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, pages 143–150, 1986. 1
4. E. Lafortune and Y. D. Willems. Bi-directional path-tracing. In *Compugraphics '93*, pages 145–153, Alvor, 1993. 8
5. M. Pauly, T. Kollig, and A. Keller. Metropolis light transport for participating media. In *Rendering Techniques*, pages 11–22, 2000.
6. J. Prikryl and W. Purgathofer. Perceptually based radiosity. In *Eurographics '98, STAR — State of the Art Report*, 1998. 8
7. Alfréd Rényi. *Wahrscheinlichkeitsrechnung*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1962. 3
8. P. Shirley, C. Wang, and K. Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996. 2
9. I. Sobol. *Die Monte-Carlo Methode*. Deutscher Verlag der Wissenschaften, 1991. 1
10. L. Szirmay-Kalos. *Photorealistic Image Synthesis with Ray-Bundles*. Hungarian Academy of Sciences, D.Sc. Dissertation, Budapest, 2000. http//www.iit.bme.hu/~szirmay.
11. L. Szirmay-Kalos, P. Dornbach, and W. Purgathofer. On the start-up bias problem of Metropolis sampling. In *Winter School of Computer Graphics '99*, pages 273–280, Plzen, Czech Republic, 1999. 3
12. E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, http://graphics.stanford.edu/papers/veach_thesis, 1997. 2, 4, 8
13. E. Veach and L. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Rendering Techniques '94*, pages 147–162, 1994. 6
14. E. Veach and L. Guibas. Metropolis light transport. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 65–76, 1997. 2, 3, 7



$p_{large} = 0.02$ $p_{large} = 0.5$ $p_{large} = 0.9$

Figure 8: *Images rendered with 25 mutations per pixel by the proposed algorithm included in bi-directional path tracing, using multiple importance sampling with different large step probabilities.*



bi-directional path tracing new method reference image

Figure 9: *Images rendered with 100 samples per pixel by bi-directional path tracing and by the proposed Metropolis method included in bi-directional path tracing ($p_{large} = 0.5$), using the same computation time. The reference image is also shown for comparison.*