Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation

Martin Šik and Jaroslav Křivánek

Abstract—Two decades have passed since the introduction of Markov chain Monte Carlo (MCMC) into light transport simulation by Veach and Guibas, and numerous follow-up works have been published since then. However, up until now no survey has attempted to cover the majority of these methods. The aim of this paper is therefore to offer a first comprehensive survey of MCMC algorithms for light transport simulation. The methods presented in this paper are categorized by their objectives and properties, while we point out their strengths and weaknesses. We discuss how the methods handle the main issues of MCMC and how they could be combined or improved in the near future. To make the paper suitable for readers unacquainted with MCMC methods, we include an introduction to general MCMC and its demonstration on a simple example.

Index Terms—Markov Chain Monte Carlo, Metropolis-Hastings, Metropolis Light Transport, Light Transport Simulation, STAR.

1 Introduction

Markov chain Monte Carlo (MCMC) [1] is a class of efficient sampling methods that have been applied in fields such as statistics, econometrics, physics, biology, linguistics, and last but not least in computer graphics. In this paper we discuss their application in a specific branch of computer graphics called light transport simulation, which concerns itself with simulating how light propagates from light sources through a virtual scene to the camera, with the purpose of synthesizing realistic images. Its applications include visual effects for movies, computer animation, architectural visualization, product design, etc.

Veach and Guibas were the first to use MCMC in light transport simulation with their Metropolis Light Transport algorithm [2] more than twenty years ago. Since then researchers have proposed numerous different MCMC methods to simulate light transport. The goals and applicability of these methods vary and it may take a substantial effort to get a high-level understanding of this research area.

For this reason the main goal of this paper is to offer a comprehensive survey of MCMC methods used in light transport (note that the survey by Jakob [3] covers only a small subset of the algorithms discussed in our work). We organize the existing algorithms into categories based on their aim and/or their common specific features. In each category, we point out strengths and weaknesses of the individual algorithms. We discuss how the existing methods handle the main issues of MCMC algorithms and how they could be combined and/or improved in the near future.

Besides the survey itself, this paper offers an introduction to general MCMC with both rigorous mathematical formulas and an easy to follow example. When describing different MCMC methods, we first discuss the particular MCMC method in general, before showing how it is used in light transport simulation. This should allow the reader to more easily differentiate between the fundamental MCMC ideas and the steps necessary for their application to graphics.

The rest of the paper is divided into three parts:

- Introduction to general MCMC including an easy to follow example and the first application of MCMC in light transport simulation (Part I).
- Comprehensive survey of light transport MCMC methods, divided by their aim and similarities into individual categories (Part II).
- Discussion and conclusion (Part III).

Part I: Introduction to MCMC and light transport simulation

We begin the first part of this survey by describing general Markov chain Monte Carlo algorithms (Sec. 2). Then we briefly introduce light transport simulation and show how it can be solved by Markov chain Monte Carlo (Sec. 3).

2 Markov chain Monte Carlo

In this section we introduce a class of Monte Carlo sampling algorithms called Markov chain Monte Carlo (MCMC) [1]. The MCMC algorithms can generate a sequence of samples distributed according to an arbitrary probability density function (which does not need to be normalized). Therefore they can be effectively used to generate samples from otherwise hard-to-sample distributions. Table 1 shows common notation used throughout this section.

2.1 General MCMC algorithm

Given any initial state u_0 from a state space[1] $\mathcal{U}$, an MCMC algorithm generates a sequence of states from $\mathcal{U}$ such that each state u_i is a realization of a random variable with a distribution which depends only on the previous state u_{i-1}.

• *Martin Šik and Jaroslav Křivánek are with the Charles University, Prague, Czech Republic*
E-mail: sik@cgg.mff.cuni.cz, jaroslav.krivanek@mff.cuni.cz

1. A state space can be e.g. a simple line of all real numbers $\mathcal{U} = \mathbb{R}$, or a subspace of a high dimensional space $\mathcal{U} \subseteq \mathbb{R}^N$. In light transport, the state space is often the space of light paths, as discussed below.

$\mathcal{U}$	Markov chain state space
$u, v \in \mathcal{U}$	Markov chain state (a sample from MCMC)
$t(u \to v)$	Transition probability of v given u
π_i^*	Distribution of possible i-th states of a Markov chain
$\pi_i^*(u_i)$	Probability density of i-th state of a Markov chain
$\pi(u)$	Target function
π^*	Stationary distribution of a Markov chain
$\pi^*(u)$	Probability density of the stationary distribution
b	Normalization constant of the target function
Q	Mutation
$q(u \to v)$	Mutation probability density of v given u
$\alpha(u \to v)$	Acceptance probability of v given u
$p(u)$	Probability of generating u given some distribution
b^*	An estimator weight used to suppress start-up bias

TABLE 1: Common notation.

This property is known as the *Markov property* and the sequence is called a *Markov chain*.

Given the Markov property, we can define a *transition function* $t(u \to v)$, which gives us the conditional probability (density) of going to state v given the current state u. Each state u_i is a realization of a random variable (sample) with some distribution, whose probability density π_i^* can be defined as an integral over all states drawn from π_{i-1}^*, weighted by the transition function

$$\pi_i^*(u_i) = \int_{\mathcal{U}} t(v \to u_i)\pi_{i-1}^*(v)\mathrm{d}v. \tag{1}$$

If we then define an arbitrary non-negative *target function* $\pi : \mathcal{U} \to \mathbb{R}_0^+$, it can be proven that under some conditions on the transition function t, the distribution π_i^* will converge to a *stationary distribution* π^* with probability density $\pi^*(u) = \pi(u)/b$ (where b is equal to the probability normalization constant $\int_{\mathcal{U}} \pi(u)\mathrm{d}u$). We can therefore consider states of the Markov chain as samples generated by MCMC from a distribution that converges to the desired distribution π^* proportional to the target function π.

For the algorithm to converge to π^*, we must impose the following conditions on the transition function t.

- The transition function t must be able to reach any state $u \in \mathcal{U}$ with positive $\pi^*(u)$ from any other state $v \in \mathcal{U}$ with positive $\pi^*(v)$ in a finite number of steps. This property is called *ergodicity*.
- We must ensure that π^* is *invariant* for the generated sequence. This means that once the sample u_i comes from the stationary distribution π^*, using transition function t to obtain the next sample u_{i+1} will result in u_{i+1} being from π^* as well.
- The transition function t must be *aperiodic*, meaning that none $u \in \mathcal{U}$ is repeated periodically.

From the first two conditions it can be proven [4] that the invariant distribution is unique and the chain is *positive recurrent*, i.e. the expected number of steps needed to visit any state $u \in \mathcal{U}$ with positive $\pi^*(u)$ is finite. The third condition then ensures that the distribution of generated samples actually converges to the invariant distribution [4].

While MCMC will converge to the given distribution, in general there is no way of telling when this will happen or how fast it will happen. We discuss this issue together with the selection of the initial sample in Sec. 2.5. In the following section, we will focus on a basic MCMC algorithm *Metropolis-Hastings* [5] and explain on it the principles in more detail. For detailed information about general MCMC, we refer the readers to a more thorough material [6].

2.2 Metropolis-Hastings algorithm

We introduce here an MCMC algorithm referred to as Metropolis-Hastings [5]. While this method is quite basic, it is the most commonly used MCMC algorithm in light transport simulation, and all other algorithms discussed in this paper are derived from it.

The pseudo-code of Metropolis-Hastings (MH) is given in Fig. 1. The algorithm starts with an initial sample (line 1), which can be selected at random from some distribution or it can be fixed. From the initial sample it continues to generate a Markov chain. Given a sample u_i the algorithm generates the next sample u_{i+1} by first generating a proposal v (line 3). The proposal v is sampled using a mutation Q with a given conditional probability (density) $q(v|u)$, more commonly denoted as $q(u \to v)$. Next, the *acceptance probability* α with which we will accept the proposal is computed (line 4). The probability is defined as

$$\alpha(u \to v) = \begin{cases} \min\left(\frac{\pi(v)q(v\to u)}{\pi(u)q(u\to v)}, 1\right) & \text{if } \pi(u) > 0 \\ 1 & \text{if } \pi(u) = 0 \end{cases} \tag{2}$$

The proposal is either accepted and set as the next sample (line 7) or rejected meaning that the current sample is repeated (line 9). An example is given in Fig. 2.

```
1: Select $u_0$
2: **for** $i = 0$ **to** the number of samples **do**
3:     Generate proposal $v$ using mutation $Q$
4:     Compute acceptance probability $\alpha(u_i \to v)$
5:     Generate random number $\xi \sim \mathrm{U}(0,1)$
6:     **if** $\alpha(u_i \to v) > \xi$ **then**
7:         Accept proposal: set $u_{i+1} = v$
8:     **else**
9:         Reject proposal: set $u_{i+1} = u_i$
```

Fig. 1: Metropolis-Hastings algorithm.

2.3 Convergence of the Metropolis-Hastings algorithm

We have already discussed that the transition function t must satisfy certain conditions, in order for the algorithm to converge. It can be shown that the transition function implied by Metropolis-Hastings indeed satisfies them.

The transition function of the MH algorithm is defined as

$$t(u \to v) = \begin{cases} q(u \to v)\alpha(u\to v) & \text{if } u \neq v \\ 1 - \int_{\mathcal{U}} q(u \to v)\alpha(u\to v)\mathrm{d}v & \text{if } u = v \end{cases} \tag{7}$$

Here, for $u \neq v$ the transition function is the probability of proposing *and* accepting the proposal v, while for $u = v$ the function value is equal to the probability of rejecting any proposed mutation (we assume for the sake of simplicity that the proposed sample differs from the current sample, i.e. $q(u \to u) = 0$, otherwise we would have to distinguish in Eq. (7) whether $u = v$ is due to rejection or acceptance).

To achieve ergodicity, one must ensure that for a given π^* the algorithm is able to reach any state $u \in \mathcal{U}$ with positive $\pi^*(u)$ via several accepted mutations. This is commonly accomplished by using a mutation that can always propose any state from $\mathcal{U}$ (e.g. the mutation Q_{U} from Fig. 2).

The invariant distribution condition (i.e. the fact that the target distribution is maintained through any transition)

Fig. 2: A simple mutation example for the Metropolis-Hastings algorithm (based on an example by Matt Pharr)

Let us consider a task of sampling according to a target function π defined as

$$\pi(u) = \begin{cases} (u-0.5)^2 & \text{if } u \in [0,1] = \mathcal{U} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In order to use the Metropolis-Hastings (MH) algorithm to generate samples according to π, we must define mutations. In the following text we assume the algorithm is always initialized with $u_0 = 0$ for illustrative purposes.
We first define a simple uniform mutation Q_{U}

$$Q_{\mathrm{U}}(u, \xi) = \xi, \tag{4}$$

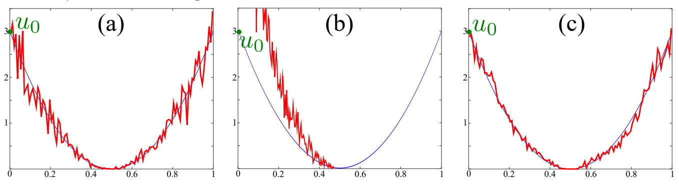
where $\xi \sim \mathrm{U}(0,1)$ is a uniformly distributed random number and u is the current sample. The mutation Q_{U} is clearly independent on the current sample and generates each $v \in \mathcal{U}$ with the same probability. Since Q_{U} ensures that it is possible to visit every $v \in \mathcal{U}$ for which $\pi(v) > 0$ in just one step, the algorithm will converge. Histogram (a) shows samples generated using Q_{U}.
We now define another mutation Q_{L}

$$Q_{\mathrm{L}}(u, \xi) = u + \frac{\xi - 0.5}{10}. \tag{5}$$

Unlike the previous mutation, this one is dependent on the current sample u and generates proposals only in a small vicinity of u. Still, in a finite number of accepted steps it can visit every $v \in \mathcal{U}$ for which $\pi(v) > 0$ and therefore the algorithm will eventually converge. Histogram (b) shows a result of using Q_{L} to generate samples from π. While the algorithm should theoretically converge, we see that no $v > 0.5$ was generated. This is caused by poor *global exploration*, meaning the algorithm failed to discover all islands of the state space $\mathcal{U}$ where $\pi(v) > 0$.
Let us now consider a combined mutation Q_{C}

$$Q_{\mathrm{C}}(u, \xi_1, \xi_2) = \begin{cases} Q_{\mathrm{U}}(u, \xi_2) & \text{if } \xi_1 < 0.1 \\ Q_{\mathrm{L}}(u, \xi_2) & \text{otherwise} \end{cases} \tag{6}$$

where both ξ_1 and ξ_2 are uniform random numbers. As can be seen from the histogram (c), using such a combined mutation will get us superior results than just using Q_{L} or Q_{U}. This is thanks to the mutation Q_{L} ensuring good *local exploration* (i.e. good sampling of local subspaces of $\mathcal{U}$, also known as *exploitation*), while Q_{U} ensures good *global exploration* (i.e. discovery of the important subspaces of $\mathcal{U}$). *A good balance between local and global exploration is generally the key to a good performance of all algorithms based on Metropolis-Hastings.*



This figure shows normalized target function (blue) and histograms of 20,000 samples (red) generated by ten runs of Metropolis-Hastings with different mutations: (a) Uniform mutation Q_{U}, (b) local mutation Q_{L} and (c) combination of both Q_{C}. All runs are initialized with the same $u_0 = 0$.

can be proven [7] from the fact that Metropolis-Hastings satisfies the so called *detailed balance* for all $u, v \in \mathcal{U}$:

$$\pi(u)q(u \to v)\alpha(u \to v) = \pi(v)q(v \to u)\alpha(v \to u). \tag{8}$$

It is important to note that while the *detailed balance* is a sufficient condition, it is not a necessary one (see Sec. 2 of the supplemental material).

Aperiodicity of the transition function is ensured if ergodicity holds and at least one state $u \in \mathcal{U}$ has non zero probability of being repeated ($t(u \to u) > 0$) [8].

2.4 Using MCMC for quadrature

One of the common uses of MCMC is for numerical integration (quadrature), which is also the case of all light transport algorithms presented here. Let us first consider the following integral

$$I = \int_{\mathcal{U}} h(u)\pi(u)\mathrm{d}\mu(u), \tag{9}$$

where π is a non-negative function and h is an arbitrary function. We can use Monte Carlo to estimate this integral as follows

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N} \frac{h(u_i)\pi(u_i)}{p(u_i)} \tag{10}$$

where the samples u_i were generated with probability density $p(u)$. If $p(u) > 0$ whenever $h(u)\pi(u) > 0$ it can be shown that the estimator is unbiased, i.e. $\mathbb{E}[\langle I \rangle] = I$. The variance of this estimator will depend on the number of samples N and on how similar the pdf p is to $h \cdot \pi$. In the ideal case, where p is exactly proportional to $h \cdot \pi$ up to a normalization constant, we get the so called *zero-variance* estimator (and thus no variance). Unfortunately, it is usually impossible to directly generate samples from such a pdf.

Let us now consider how we could utilize MCMC in this example. In MCMC settings, we can use $\pi(u)$ as the target function. This way the MCMC algorithm generates samples according to $p = \pi/b$, where $b = \int_{\mathcal{U}} \pi(u)du$ is the normalization constant. The resulting estimator is as follows

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N} \frac{h(u_i)\pi(u_i)}{p(u_i)} = \frac{b}{N}\sum_{i=1}^{N} h(u_i). \tag{11}$$

If the function h does not introduce too much variance, this estimator should theoretically have low variance.

Unfortunately, we have to compute the normalization constant b and we cannot use samples generated according to $p = \pi/b$ to do this, since the resulting estimator would be equal to the unknown b. This is also the reason why MCMC is not used to sample from $h \cdot \pi$. We have to compute b from samples generated from a distribution with a known normalization (or use other quadrature technique). Such a computation will suffer from variance and thus we may have gained nothing compared to ordinary Monte Carlo.

The advantage of MCMC will be more apparent in a scenario when we need to compute many correlated integrals, which all share the same scaling factor b. Consider a large number of integrals that differ only in the $h(u_i)$ function:

$I_1 = \int_{\mathcal{U}} h_1(u)\pi(u)\mathrm{d}\mu(u), I_2 = \int_{\mathcal{U}} h_2(u)\pi(u)\mathrm{d}\mu(u), \ldots$ In that case we can utilize the same samples generated according to π to estimate all of the above integrals and, more importantly, their common factor b is computed just once. This is exactly the case of light transport simulation (pixel values are the individual integrals, as will be discussed in Sec. 3.1).

2.5 Start-up bias

While a Markov chain steadily converges to the stationary distribution, it will only reach it in the infinity. Therefore especially in the beginning, the distribution of Markov chain samples can be far from the stationary distribution. This issue is often referred to as *start-up bias* [9]. A common solution applied in computational statistics is to throw away a number of samples [6]. However, it is unclear how many of these samples are sufficient, in order to get the start-up bias below some threshold.

Veach and Guibas [2] proposed a different solution for handling start-up bias when computing quadrature. The solution requires an alternative sampling technique (like bidirectional path tracing [10] in the case of light transport simulation) that selects an initial state of the Markov chain. If the initial state u_0 were selected with probability equal to $p_0(u_0)$, the quadrature estimate would be just weighted with $\frac{\pi(u_0)}{p_0(u_0)}$ instead of the normalization constant b. It can then be proven [2] that an average over many estimates of the quadrature will be unbiased.

Unfortunately, each individual estimate can be completely off. Just consider a situation when $\pi(u_0)$ equals zero, in that case the estimate will be zero as well. Veach and Guibas address this issue by randomly selecting a whole set of N initial samples $\{u_{0,i}; i = 1, \ldots, N\}$. Among these samples the one initial sample is selected by random selection with probabilities proportional to the weights $\frac{\pi(u_{0,i})}{p_0(u_{0,i})}$. The quadrature is then weighted by the following weight

$$b^* = \frac{1}{N}\sum_{i=1}^{N}\frac{\pi(u_{0,i})}{p_0(u_{0,i})}. \tag{12}$$

Note that b^* is a Monte Carlo estimate of the normalization $b = \int_{\mathcal{U}} \pi(u)\mathrm{d}u$, that is $\mathbb{E}[b^*] = b$. Since this technique for removing start-up bias is applied in almost all algorithms discussed in this paper, from now on we will use the weight b^* instead of the true normalization b in all equations.

2.6 Common optimization

Before we move on to light transport, we give here a few tips for optimizations used in MCMC quadrature computation.

Parallelization. First, modern hardware is able to run several threads at once, therefore it is advantageous to run MCMC algorithm in parallel. Effectively parallelizing a generation of one Markov chain is a difficult task (although it is possible [11]). Instead, several independent chains are commonly computed in parallel and their estimates are combined. When initializing the chains, one can draw their individual initial samples from the same set of initial samples, thus their quadrature estimates will be weighted by the same constant b^*.

Convergence testing. Using several independent chains has another advantage: we can compare their estimates in order to determine whether the overall quadrature estimate has converged enough. We should be careful though, since having similar estimates does not always ensure the algorithm has sampled all important parts of the state space properly due to poor global exploration (Fig. 2).

The use of expected values. When we use MCMC to compute quadrature, we can use the rejected proposals to improve the estimation. Common MCMC algorithms (like MH) use a step that includes accepting or rejecting a proposed sample v with a given probability $\alpha(u \to v)$ (where u is the current sample). Therefore the proposal v contributes to the quadrature with probability $\alpha(u \to v)$, while the current sample u contributes with probability $1 - \alpha(u \to v)$. We can thus improve efficiency by always accumulating both samples u and v weighted by the corresponding probabilities. Effectively, this optimization replaces a random variable by its expected value [7, p. 357]. This technique is an example of Rao-Blackwellization [12], and it is especially useful for parts of the state space where the target function value is low and therefore fewer samples are distributed there.

3 MCMC IN LIGHT TRANSPORT

In this section we show how to use MCMC as a sampler for Monte Carlo light transport simulation. We first introduce light transport equation in Sec. 3.1 and show how it can be computed using general Monte Carlo. We also discuss in Sec. 3.2 a common technique for combining estimators called *Multiple importance sampling* (MIS) [13], since it is used in many of the algorithms described later.

Once we understand the light transport equation, we demonstrate on an example how we can apply MCMC to solve it. As the example algorithm we have chosen the original Veach and Guibas' Metropolis light transport (MLT) [2], since it was this work that introduced MCMC to light transport. The methods described in the subsequent sections then attempt to improve upon the original MLT algorithm.

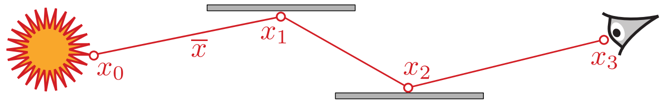


Fig. 3: A light transport path $\overline{x}$ can be imagined as a polyline between a light source and the camera. The interior vertices correspond to light interactions with scene surfaces.

3.1 Light transport equation

Light transport can be expressed by the path integral

$$I_j = \int_{\Omega} h_j(\overline{x}) f(\overline{x}) \mathrm{d}\mu(\overline{x}), \tag{13}$$

where I_j is the pixel value of the j-th pixel, Ω is the space of all possible light paths. A light transport path $\overline{x} \in \Omega$ can be imagined as a polyline that represents the trajectory of a packet of light energy traveling through the scene. The first vertex lies on a light source, the interior vertices correspond to light reflection/refraction or scattering, while the last vertex is at the camera (Fig. 3). A path of length k can therefore be represented as a vector of $k+1$ vertices $\overline{x} = (x_0, \ldots, x_k)$. The path contribution function f gives the amount of light energy transported along the path $\overline{x}$. h_j is the pixel filter of the pixel j and μ is the measure associated with the path space Ω [7].

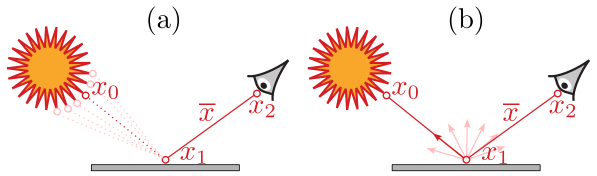


Fig. 4: We can construct the same light path $\overline{x}$ in various ways. Given already sampled vertices x_1 and x_2, the vertex x_0 can be sampled by (a) selecting x_0 on a light source or (b) sampling direction from x_1 and intersecting the light by a corresponding ray. A path sampling technique defines how each vertex of a path is sampled, and has an associated probability density function (pdf). See other sources for more details about path sampling [14, p. 1003].

Given a *path sampling technique* (e.g. path tracing, see Fig. 4, or light tracing or any of the techniques used in bidirectional path tracing [10]) that generates N random paths $\overline{x}_i$ according to the probability density function (pdf) $p(\overline{x}_i)$, we can estimate the path integral using a general Monte Carlo estimator

$$\langle I_j \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\overline{x}_i) f(\overline{x}_i)}{p(\overline{x}_i)}. \tag{14}$$

Note that a different estimate is computed for each pixel of the image (indexed by j). Since the pixel integrals all share the path contribution function f, using MCMC to compute all estimates simultaneously is advantageous (they all have the same normalization factor b, see Sec. 2.4).

3.2 Multiple Importance Sampling

As per the importance sampling principle, a path sampling pdf roughly proportional to the path contribution function f reduces the variance of the estimator in Eq. (14). While it may be difficult to find such a pdf, we can often construct multiple path sampling techniques with pdfs that approximate f locally. These path sampling techniques differ in the way they sample different vertices of the path (Fig. 4). We can then reduce the variance if we combine path sampling techniques into one weighted sum

$$\langle I_j \rangle = \sum_{m \in \mathcal{M}} \frac{1}{N_m} \sum_{i=1}^{N_m} w_m(\overline{x}_{m,i}) \frac{h_j(\overline{x}_{m,i}) f(\overline{x}_{m,i})}{p_m(\overline{x}_{m,i})}. \tag{15}$$

Here we estimate the pixel value I_j using a set of path sampling techniques $\mathcal{M}$. A technique m generates N_m samples $\overline{x}_{m,i} \propto p_m(\overline{x}_{m,i})$. The multiple importance sampling (MIS) weight $w_m(\overline{x}_{m,i})$ of the technique m for the path $\overline{x}_{m,i}$ is an arbitrary positive function. The weights across the different techniques must sum up to one ($\sum_{m \in \mathcal{M}} w_m(\overline{x}_{m,i}) = 1$) to ensure unbiased estimation. For an ordinary Monte Carlo estimator, setting the weight according to the *balance heuristic* results in close-to-optimal variance [13].

As an example, the bidirectional path tracing algorithm [10] employs MIS to combine path sampling techniques corresponding to generating different-length subpaths from lights and the camera, respectively.

3.3 Metropolis Light Transport

Veach and Guibas have introduced MCMC to computer graphics [2] in their algorithm *Metropolis light transport*

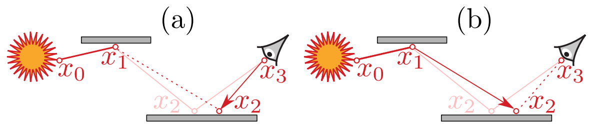


Fig. 5: In Metropolis light transport (Sec. 3.3), the lens perturbation (a) creates a new path by changing the direction from a selected vertex x_3 towards the next vertex closer to a light source x_2, the vertex x_1 is then connected to x_2. The caustic perturbation (b) works in the opposite way by changing the direction towards the camera from a vertex closer to a light source x_1 and connecting x_3 to a new vertex x_2.

(MLT). MLT uses Metropolis-Hastings (Sec. 2.2) and applies it directly in the path space (i.e. state space of the Markov chain is equal to the path space, one state corresponds to one full light transport path). The target function is proportional to the scalar luminosity of the light path contribution to the image $f^*(\overline{x}) = lum(f(\overline{x}))$. This leads to the following estimator of the light transport equation

$$\langle I_j \rangle = \frac{Pb^*}{N} \sum_{i=1}^{N} \frac{h_j(\overline{x}_i) f(\overline{x}_i)}{f^*(\overline{x}_i)}, \tag{16}$$

where P is the number of pixels and the start-up weight b^* (see Sec. 2.5) is computed from the set of initial samples generated by bidirectional path tracing [10].

Veach and Guibas have devised three different local mutations (which they refer to as *perturbations*). The *lens* perturbation can be effectively used for perturbing paths containing specular surfaces directly visible from the camera, while the *caustic* perturbation is more suited for caustics (i.e. path where light is concentrated via specular reflections before hitting a diffuse surface directly visible from the camera). Fig. 5 illustrates lens and caustic perturbations. Finally, the *multi-chain* perturbation performs lens and caustic perturbations at once in order to handle difficult to sample paths (such as reflected caustics).

The ergodicity is ensured by the *bidirectional* mutation, which effectively replaces any subpath of the current path. To improve stratification on the image plane, MLT uses the *lens-subpath* mutation which changes the path by starting it on a different pixel. MLT was later improved [15] to better handle participating media and two new mutations were introduced: one that perturbs the path by changing the scattering direction and another that changes the length of a path segment in a medium. Fig. 6 compares MLT against a popular classic Monte Carlo method.

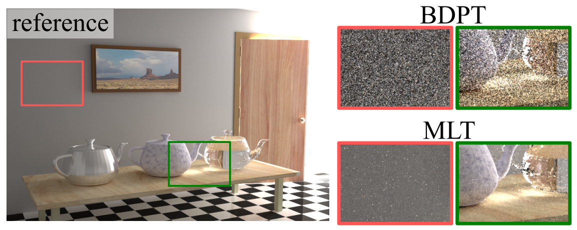


Fig. 6: Equal-time comparison (12 min.) of MLT (Sec. 3.3) and bidirectional path tracing (BDPT) [10]. In this case, MLT produces much less noise thanks to its ability to generate more paths that pass through the door ajar and can thus contribute to the image. Scene courtesy of Lehtinen et al. [16].

PART II: MCMC ALGORITHMS FOR LIGHT TRANSPORT SIMULATION

Before we discuss the individual MCMC methods, let us briefly summarize important aspects of an MCMC algorithm effective at light transport simulation. First, it is important to ensure that once MCMC sampled a path, it can then generate similar paths to effectively sample a local area of the path space. This property is called **local exploration**. However, excessive local exploration results in high sample correlation, which is then visible in the image as groups of overly-bright pixels.

To avoid this issue, an MCMC algorithm should be able to quickly discover other contributing areas of the path space. This **global exploration** is enabled by proposing paths further away from the current path. However given the usual target function with many local maxima/discontinuities, if such a path is not selected carefully, it will likely have lower contribution and thus lower chance of acceptance (Eq. 2). This will result in the algorithm "getting stuck" in one mode of the target function, while completely failing to discover other modes. The resulting image may thus miss much of the light transport (Fig. 7).

In light transport, multiple integrals are estimated at once, thus it is important for an MCMC algorithm to ensure **uniform error** reduction. This way the user can clearly see how the image estimation progresses. The methods described in the following text are divided into categories based on which MCMC aspect they try to improve and how they approach that goal.

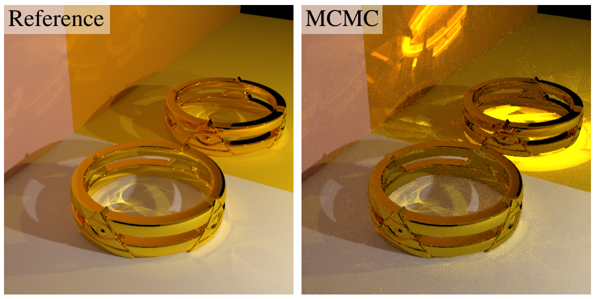


Fig. 7: The image rendered by an example MCMC algorithm with poor global exploration (right) contains overly-bright regions of pixels. They are caused by the MCMC chain getting stuck in narrow peaks of the target function.

4 SIMPLIFIED STATE SPACE

We have described the Metropolis light transport (MLT) algorithm that directly uses the path space Ω as the Markov chain state space. While mutating paths directly in the path space can be effective, it also results in a rather complex algorithm that has to rely on a number of mutation strategies, each fine-tuned to handle different transport features.

In this section we describe algorithms that use a simplified state space. Not only does it lead to simpler algorithms, it also allows to use a simple mutation that is effective at *local exploration* of many types of paths, while improving *global exploration* as well.

4.1 Primary Sample Space Metropolis Light Transport

Primary sample space MLT (PSSMLT) [17] by Kelemen et al. utilizes the fact that each light path $\overline{x}$ is uniquely defined by a vector of random numbers $\overline{u} \in \mathcal{P} = [0,1]^{O(k)}$ (i.e. given a path sampling technique, a path of length k can be generated using $O(k)$ random numbers). Unlike the original MLT, which mutates light paths directly in the path space, path mutation in PSSMLT is achieved by mutating these random vectors $\overline{u}$ and then mapping them to the path space. The mapping $m : \mathcal{P} \to \Omega$ is performed by sampling path vertices using the $\overline{u}$ as random numbers (Fig. 8) given a fixed path sampling technique (Fig. 4).

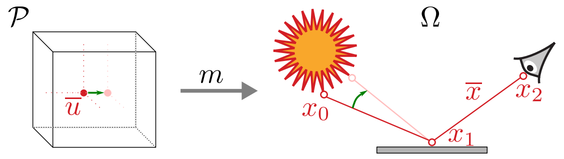


Fig. 8: A random vector $\overline{u}$ from the primary sample space $\mathcal{P}$ is mapped to a path $\overline{x}$ in the path space. The mapping m is done by sampling vertices of $\overline{x}$ with a given path sampling technique using $\overline{u}$ as random numbers (see e.g. [14, p. 1003] for more details on path sampling). Changing $\overline{u}$ (green arrow) results in a different path $\overline{x}$ being sampled.

4.1.1 Estimating the light transport equation

In order to solve the light transport equation using PSSMLT, the authors start by making a substitution in the path integral (Eq. (13)) that changes the integration domain from the path space Ω to the primary sample space $\mathcal{P}$

$$\begin{aligned} I_j &= \int_\Omega h_j(\overline{x}) f(\overline{x}) \mathrm{d}\mu(\overline{x}) \\ &= \int_{\mathcal{P}} \frac{h_j(m(\overline{u})) f(m(\overline{u}))}{p_m(m(\overline{u}))} \mathrm{d}\overline{u}. \end{aligned} \tag{17}$$

The change of variable requires the Jacobian $|\mathrm{d}\mu(\overline{x})/\mathrm{d}\overline{u}| = 1/p_m(m(\overline{u}))$ of the mapping m, which corresponds to the inverse of the probability density of sampling the path $\overline{x}$ using the sampling technique m. If we define $C(\overline{u}) = \frac{f_m(m(\overline{u}))}{p_m(m(\overline{u}))}$, Eq. (17) can be estimated using MCMC as follows

$$\langle I_j \rangle = \frac{Pb^*}{N} \sum_{i=1}^{N} \frac{h_j(m(\overline{u}_i)) C(\overline{u}_i)}{C^*(\overline{u}_i)}, \tag{18}$$

where the target function C^* is the scalar luminosity of C. The start-up weight b^* is computed as discussed in Sec. 2.5

4.1.2 Mutations

Only the random vectors $\overline{u}$ are mutated in PSSMLT (the paths change as a consequence). Two simple mutations are used: *Small step*, which handles local exploration by slightly perturbing the entire random vector $\overline{u}$ and *large step* which ensures ergodicity by generating a new independent vector $\overline{u}$. The small step is implemented by independently perturbing each component of the vector $\overline{u}$ representing the current Markov Chain state. The perturbation relies on a distribution with exponential falloff centered at the current state. Both mutations are chosen to be symmetric and thus the mutation probability densities $q(v \to u)$ and $q(u \to v)$ are equal and they cancel out in the acceptance equation (Eq. (2)).[2] The mutation type (small, large) is selected randomly at each step with a probability either given by the user or automatically tuned [18].

2. In the original MLT the mutations are not symmetric and therefore the mutation probability must be always computed.

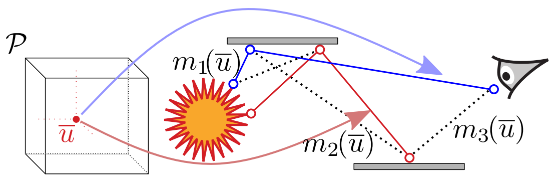


Fig. 9: Primary sample space MLT (Sec. 4.1) with many mappings treats $\overline{u}$ as a vector of random numbers to generate a camera subpath (blue) and a light subpath (red). The subpaths are combined (dashed line) in various ways yielding several full paths. Each combination corresponds to one mapping m_i.

In the original MLT different type of paths require different type of mutations to ensure efficiency, while in PSSMLT the mutation effectiveness depends mostly on how well the mapping m distributes paths to the path space. The mapping (path sampling) is often performed by local importance sampling at each vertex and thus the mutations are good at local importance sampling, but are not optimal for non-local features (e.g. a chain of highly glossy interactions). Note that due to importance sampling the large step is better at global exploration compared to the bidirectional mutation of the original MLT.

4.1.3 Utilizing many mappings

We can combine several sampling techniques using MIS (Sec. 3.2) in order to reduce estimator variance. The same principle can be applied in PSSMLT. Each sampling technique corresponds to a different mapping from the primary sample space to the path space, and thus given a set of path sampling techniques we can map one sample $\overline{u}$ to several different paths. PSSMLT relies on MIS to combine the contributions of all paths created from a single sample $\overline{u}$

$$C(\overline{u}) = \sum_{m \in \mathcal{M}(\overline{u})} w_m(m(\overline{u})) \frac{f(m(\overline{u}))}{p_m(m(\overline{u}))}, \tag{19}$$

where $\mathcal{M}(\overline{u})$ are all the available mappings for a given random vector $\overline{u}$ and w_m is the MIS weight. With this modification of the definition of $C(\overline{u})$, the final pixel estimator in Eq. (18) remains the same.

The use of multiple mappings decreases the total variation of the target function C^*, which in turn means the proposed samples will be less often rejected and the sampler will be more efficient. On the other hand, constructing several paths from each sample can be costly.

PSSMLT usually defines its mappings using sampling techniques from bidirectional path tracing [10]. This means that a camera subpath (generated from the camera) and a light subpath (generated from a light source) is generated separately using the random numbers $\overline{u}$ and then they are combined in various ways yielding several full paths. Each combination then corresponds to one mapping (Fig. 9). All paths generated from a single $\overline{u}$ are then treated as one sample and are thus accepted or rejected together.

Note that primary sample space has also been used in different scenarios, e.g. in a finite-element global illumination algorithm [19].

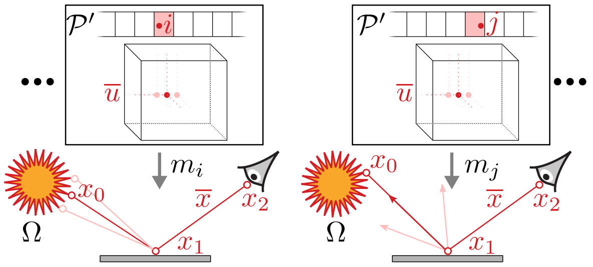


Fig. 10: Multiplexed MLT (Sec. 4.2) uses the *extended* primary sample space $\mathcal{P}'$, where the first dimension is used to determine a mapping from the rest of $\mathcal{P}'$ to the path space Ω. Here two mappings m_i and m_j map $\overline{u}$ to different paths.

4.2 Multiplexed Metropolis Light Transport

Multiplexed Metropolis light transport (MMLT) [20] improves on PSSMLT by using just one mapping for each sample. The mapping is sampled randomly as a part of the MCMC mutation in such a way that mappings generating higher-throughput paths are preferred.

As in PSSMLT, the state space in MMLT is a unit cube of random numbers. However, here the first component of $\overline{u}$ is used to determine the mapping, while the remaining components are actually mapped to the path space (Fig. 10).

4.2.1 Sampling the mapping

In order to efficiently sample the mapping m_i, MMLT modifies the PSSMLT target function (Sec. 4.1.1) by including the MIS weight

$$C_i^*(\overline{u}) = w_{m_i}(m_i(\overline{u})) \frac{f^*(m_i(\overline{u}))}{p_{m_i}(m_i(\overline{u}))}. \tag{20}$$

Since the MIS weight w_{m_i} is now part of the target function, MCMC will distribute more samples to mappings with higher w_{m_i}, which are exactly those that are more effective at sampling the given path type. The MMLT estimator differs from PSSMLT (Eq. (18)) by using just one mapping per primary sample $\overline{u}$. Furthermore, we need to multiply by the number of available mappings $\mathcal{M}(\overline{u})$.

4.2.2 Comparison to PSSMLT

MMLT prefers mappings that generate paths with high contribution, unlike PSSMLT which always uses all possible mappings. Imagine an extreme case, where in PSSMLT one sample is mapped to many very long paths with zero contribution and one short path that has a high contribution. The target function value for such a sample will be high and thus the MCMC algorithm will then generate similar samples and waste time by mapping them to many long paths with possibly no contribution.

On the downside, MMLT cannot effectively apply *Russian roulette* (RR) [7]. RR is a method to terminate sampling new path vertices if the current path throughput is below certain threshold, while ensuring the estimator remains unbiased. When RR is applied in PSSMLT, the subpaths generated from a sample are possibly shorter and thus the various mappings yield fewer paths. In MMLT, one selects a mapping a priori and thus determines subpaths' length before generating them. RR can therefore only reject a whole subpath rather than shorten it. Fig. 11 shows a comparison of PSSMLT, MMLT and the original MLT.

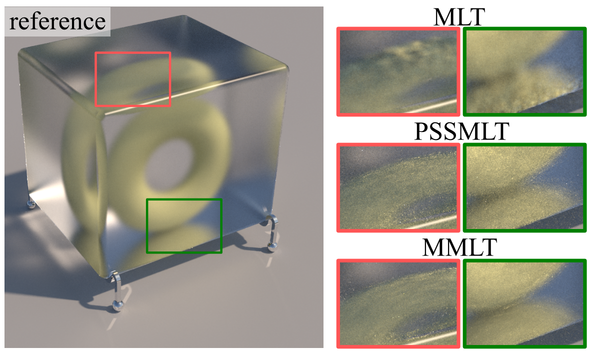


Fig. 11: The original MLT (Sec. 3.3) effectively mutates some types of paths, however paths with many specular/glossy interactions are handled poorly. Primary sample space MLT (PSSMLT, Sec. 4.1) maps its samples to the path space using numerous mappings and thus PSSMLT handles effectively more types of paths than MLT. Multiplexed MLT (MMLT, Sec. 4.2) improves upon PSSMLT by using only one mapping per sample. All methods ran for 10 minutes. Scene courtesy of W. Jakob.

4.3 Fusing state spaces

Recently, two papers [21], [22] independently proposed *fusing of state spaces*, a framework in which the mutations from both the primary sample space and the path space can be combined in one algorithm. This allows to use the simple primary sample space mutations for efficient global exploration and local exploration of many types of paths, while also utilizing specialized path space mutations (such as those described in Sec. 5) leading to a more robust algorithm (see Fig. 12 for a comparison). Note that the framework introduced later by Bitterli et al. [23] allows fusing of state spaces, but the authors have left the actual combination of mutations from both state spaces for future work.

4.3.1 Inverse mapping

In order to utilize both kinds of mutations to generate one Markov chain, we must be able to convert paths/samples between the state spaces. Mapping from the primary sample space to the path space is defined by a given path sampling technique m_i as in PSSMLT and MMLT. To map paths from the path space back to the primary sample space, one must define an *inverse mapping* m_i^{-1} such that $m_i^{-1}(\overline{x}) = \overline{u}$ implies $m_i(\overline{u}) = \overline{x}$. As an example, consider a path tracing sampling technique that generates a new direction at each path vertex using random numbers from $\overline{u}$. Given a resulting path, we must be able to convert the sampled directions back to the vector of random numbers. Unfortunately, not all mappings are bijections, which complicates the definition of inverse mapping. However, even some non-bijective mappings can be handled as discussed in the original papers [21], [23].

4.3.2 Mutating in both spaces

Since the different frameworks for fusing state spaces are mathematically equivalent, we base our description on the work of Otsu et al. [21]. The authors build their algorithm on MMLT and thus utilizing primary sample space mutations is straightforward. To apply path space mutation, the current state $\overline{u}$ is mapped to the path space $\overline{x} = m_i(\overline{u})$ using the current mapping m_i and then $\overline{x}$ is mutated by any path space mutation. The resulting path $\overline{x}'$ is converted back to the primary sample space using the inverse mapping $\overline{u}' = m_i^{-1}(\overline{x}')$ and $\overline{u}'$ is either accepted or rejected. Note that the transition between the spaces must be reflected in the mutation probability when computing acceptance probability, see the original paper [21] for more details.

4.3.3 Improving MMLT efficiency

When MMLT switches from one mapping to another, the resulting path can often be very different from the previous one despite only a minor change of the random vector $\overline{u}$ (Fig. 10). This may lead to a low acceptance probability. One can utilize the inverse mapping to alleviate this problem [22], [23]. Consider a sample $\overline{u}$ that is mapped to the path space $\overline{x} = m_i(\overline{u})$ using the mapping m_i. When a new mapping m_j is selected, one can first convert $\overline{x}$ using the inverse mapping m_j^{-1} to the primary sample space yielding $\overline{u}' = m_j^{-1}(\overline{x})$. Mutating $\overline{u}'$ using the small step and mapping it back to the path space using m_j then results in a small perturbation of $\overline{x}$.

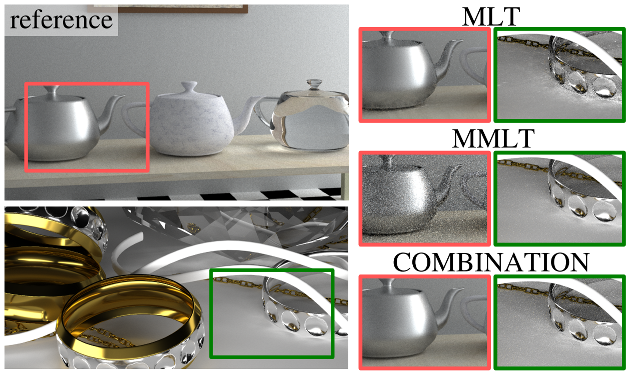


Fig. 12: Combination of MMLT and MLT mutations (Sec. 4.3) leads to a more robust algorithm, which overall performs on par with the best of the two algorithms in each scene. Image courtesy of H. Otsu et al. [21].

5 Better local exploration: Differential methods

In this section we describe methods that use differentials of scene geometry or of the target function to design new mutations effective at local exploration.

5.1 Manifold Exploration Light Transport

While the original MLT has three specialized mutations that effectively sample local subspace, they do not handle all path types equally well. Especially paths that contain chains of specular/glossy interactions are sampled poorly. To handle such paths more efficiently, Jakob and Marschner [24] introduced *Manifold exploration light transport* (MELT).

5.1.1 Specular manifold

MELT differs from MLT by its *manifold exploration* mutation, which is specifically tailored to sample effectively the so called *specular manifold*. Let us first realize that when ray hits a specular surface, the next direction is fully determined by the law of reflection or Snell's law of refraction. If a path contains a whole chain of specular interactions, changing incoming direction to the first/last vertex in the chain results

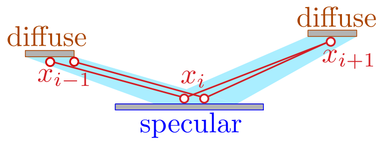


Fig. 13: The specular manifold (blue), discussed in Sec. 5.1.1, contains all subpaths created from the original subpath (red) by perturbing position of either x_{i-1} or x_{i+1}, while keeping the constraint c_i^S (Eq. (21)) on the specular vertex x_i equal to zero, corresponding to fulfilling the law of reflection.

in a deterministic change of positions of all of the specular interactions.

More formally, we can associate with each specular interaction a constraint that involves its position and the position of the preceding and following vertices:

$$c_i^S(x_{i-1}, x_i, x_{i+1}) = 0. \tag{21}$$

This constraint is equal to zero only if the generalized half-vector [25] defined by these vertices is aligned with surface normal, which corresponds to obeying the law of reflection/refraction. Given a subpath $\overline{x}$ of $k+2$ vertices that contains a chain of k specular vertices and starts and ends with non-specular vertices (x_0 and x_{k+1}, respectively) we can define a specular manifold as

$$\mathcal{S} = \{\overline{x} | c_1^S(x_0, x_1, x_2) = 0 \wedge \ldots \wedge c_k^S(x_{k-1}, x_k, x_{k+1}) = 0\}.$$

An example of a specular manifold is shown in Fig. 13.

5.1.2 Manifold mutation

To explore the specular manifold $\mathcal{S}$ the authors propose the so called *manifold walk*, where one endpoint of a subpath is moved to a new location and the specular vertices are moved accordingly so the constraints hold. To achieve this, manifold walk works in iterations which include computing geometry differentials and raytracing, thus the algorithm is quite computationally expensive.

Using the manifold walk, a mutation can be performed for a subpath with three non-specular vertices x_a, x_b, x_c as follows (Fig. 14). Sample a new direction from x_a, update specular vertices up to a new position of x_b. The rest of the vertices up to x_c is then updated using the manifold walk. Fig. 15 shows how the manifold exploration mutation can improve the original MLT algorithm.

To preserve detailed balance Eq.(8) one must ensure that the mutation is reversible. For this reason, after the path has been updated using the manifold walk, the manifold walk is executed again, but in the opposite direction (i.e. x_b is moved to its original location). If the reverse manifold

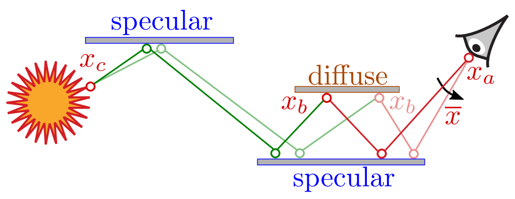


Fig. 14: The manifold exploration mutation (Sec. 5.1.2) first changes the direction from x_a and then raytraces the path up to the first diffuse vertex x_b. The manifold walk then updates all specular vertices after x_b up to x_c (green).

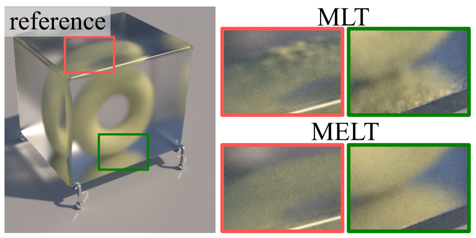


Fig. 15: The original MLT (Sec. 3.3) does not handle paths with many glossy/specular vertices effectively. Manifold exploration light transport (MELT, Sec. 5.1) adds a new mutation to MLT that is effective at generating such paths. Both methods were run for 10 minutes. Scene courtesy of W. Jakob.

walk fails or converges to a different than the original configurations (due to discontinuities), the whole mutation must be rejected. This step, along with the high cost of the manifold walk, makes the mutation much more time consuming than standard MLT mutations.

The authors also modify the mutation to handle paths with glossy vertices (instead of specular ones). While they show that the modified mutation works well on glossy surfaces compared to the original MLT mutations, there is still room for improvement as discussed next.

5.2 Half-vector Space Light Transport

Kaplanyan et al. [26] introduced the *Half-vector space light transport* (HSLT) algorithm, which builds upon the original MLT. Unlike the manifold exploration, it allows effective sampling of all manifolds (not just specular ones) by introducing the *natural constraints space* (NCS).

5.2.1 Natural constraints space

The natural constraints space is based on the fact that every light path $\overline{x} = (x_0, \ldots, x_k)$ can be represented by its end vertices (x_0, x_k) and by a projected half vector $h_i^{\perp}$ (a half-vector h_i projected to the tangent plane at x_i) instead of each inner vertex. A path $\overline{h}$ in the NCS is defined as

$$\overline{h} = (x_0, h_1^{\perp}, \ldots, h_{k-1}^{\perp}, x_k) \in \Omega(\overline{h}) \subset \Omega. \tag{22}$$

The manifold $\Omega(\overline{h})$ is a subspace of the path space in which each path is uniquely represented by the half vectors at the inner vertices (Fig. 16).

Operating with paths in the NCS has two major advantages compared to using the classical representation by a sequence of vertex positions. First, all interactions (specular, glossy, and diffuse) can be treated in the same way (i.e. there is no need to treat specular interactions differently). Second, the path integral in the NCS is decomposed into weakly

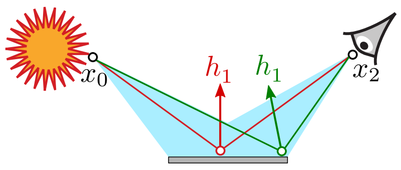


Fig. 16: The blue area represents the manifold that contains all paths with the end vertices x_0 and x_2 that are uniquely represented by the half-vectors (h_1).

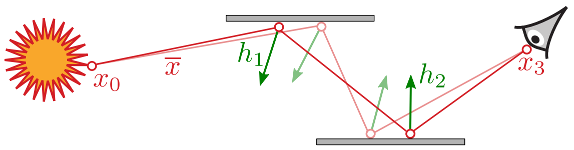


Fig. 17: Half-vector space mutations (See 5.2.2) perturb all half-vectors (green) of inner vertices at once. This results in a change of position of all inner vertices.

connected 2D integrals at each half-vector. Therefore each half-vector can be perturbed independently and the change in the path contribution can be predicted from the local change reducing the effective dimensionality of the path integral in this space.

5.2.2 Half-vector space mutation

The HSLT mutation takes a path $\overline{h}$ in the NCS and perturbs all the half-vectors resulting in $\overline{h}'$ (Fig. 17). Once the half-vectors are perturbed, an iterative algorithm similar to the manifold walk determines the actual vertex positions corresponding to the perturbed half vectors $\overline{h}'$ in NCS.

As in MELT, the iteration may fail due to discontinuities (i.e. if the local neighborhood is not smooth) or if the mutation cannot be reversed. The authors point out that HSLT is more sensitive to these discontinuities than manifold exploration. Thus to improve efficiency when the reversibility check fails, the path is not rejected but instead it is proposed as a result of a mutation that jumps between two manifolds. Fig. 18 shows a comparison of HSLT and MELT.

5.2.3 Improved HSLT

The HSLT mutation was later improved [27] in two important aspects. First, instead of using half-vectors projected to the tangent space $h^{\perp}$, the improved algorithm uses plane-to-plane projection of half-vectors $h^{\parallel}$ (Fig. 19). This allows better sampling of the perturbation, since $h^{\parallel}$ is valid in all $\mathbb{R}^2$ (unlike $h^{\perp}$ which requires clamped distributions) and corresponds to the domain used for sampling microfacet-based bidirectional reflection functions.

Second important contribution is the improvement of HSLT performance in scenes with many geometric discontinuities (such as finely tessellated and/or displaced objects). The improved HSLT breaks the path to subpaths that are perturbed separately in the NCS. This break down is done at carefully selected vertices in order to avoid

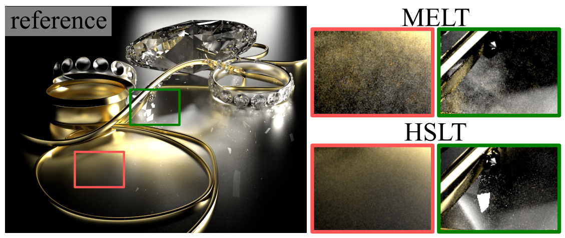


Fig. 18: Equal time comparison (30 min) of Manifold exploration light transport (MELT, Sec. 5.1) and Half-vector space light transport (HSLT, Sec. 5.2)) demonstrates that the HSLT mutation is better at local exploration of glossy paths. Scene courtesy of Kaplanyan et al. [26].

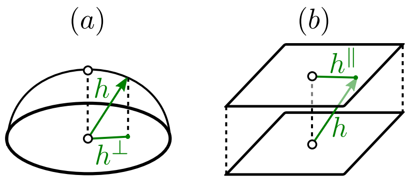


Fig. 19: Half-vector h can be mapped to a plane using either (a) projection to the tangent plane ($|h^{\perp}| \in [0,1]$) or (b) plane-to-plane projection ($|h^{\parallel}| \in [0,\infty)$).

iterating/walking over geometrical discontinuities (which often leads to reversibility check failure).

5.3 Hamiltonian Monte Carlo

While the methods so far discussed in this section used the path space geometry to improve local exploration, we can approach the same objective using a general MCMC algorithm *Hamiltonian Monte Carlo* (HMC) [28]. HMC adapts the mutations according to derivatives of the target function and thus has the potential to achieve better local exploration (the previous methods only used geometry differentials). Furthermore, HMC can lead to higher acceptance rate compared to Metropolis-Hastings thus reducing the chance of a chain getting stuck in local maxima. In the following text, we describe the algorithm that introduced HMC to light transport simulation.

5.3.1 Hessian-Hamiltonian Monte Carlo

To ensure effective local exploration, the general HMC algorithm generates proposals by simulating Hamiltonian dynamics of a point mass (equivalent of the current state) over the landscape of the graph of the target function (see Sec. 3.1 of the supplemental material for more details). Li et al. [29] point out that fully simulating Hamiltonian dynamics in light transport algorithm would require costly numerical integration that would likely involve raytracing a new path at each integration step and thus would be unfeasible. Therefore they choose a different approach in their *Hessian-Hamiltonian Monte Carlo* (H^{2}MC) algorithm.

To avoid the costly integration, H^{2}MC locally approximates the target function using Taylor series while utilizing automatic differentiation [30] (see Sec. 3.2 of the supplemental material). This approximation then allows Hamiltonian dynamics to be solved analytically.

The authors show that this analytical solution of Hamiltonian dynamics results in the proposal distribution being a Gaussian that captures local features of the path space and thus allows for efficient sampling of the local subspace (see

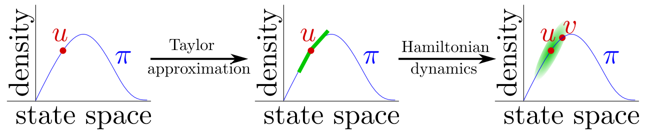


Fig. 20: In Hessian-Hamiltonian Monte Carlo (H^{2}MC, Sec. 5.3.1), the target function π (left) is locally approximated using a Taylor series (green line, middle) around the current sample u. Hamiltonian dynamics are computed using this approximation and result in a Gaussian distribution (green ellipse, right) that is used to generate a proposal v.

Fig. 20). The distribution is used to generate new proposals in the primary sample space, which are mapped like in MMLT to the path space (Sec. 4.2).

It is important to point out that while HMC was used in the derivation of the proposal distribution, the resulting algorithm uses standard Metropolis-Hastings and differs from MMLT mainly by using the anisotropic mutation instead of the isotropic small step mutation. This also means that the acceptance rate is usually lower than in the full HMC algorithm. Fig. 21 shows a comparison of H^{2}MC with MELT and HSLT. Compared to the previous two methods, the H^{2}MC mutation is usually cheaper to compute (it does not require iterative raytracing of vertices) and it also adapts to the derivatives of the target function, not just geometry. However, since H^{2}MC works in the primary sample space, it lacks the information about geometry and thus it can underperform on highly glossy or specular paths, where it is important to mutate with respect to all narrow constraints along the path. Furthermore, H^{2}MC requires the Taylor approximation of the target function, which may be difficult to achieve in practice (automatic differentiation is not always feasible, especially in production renderers).

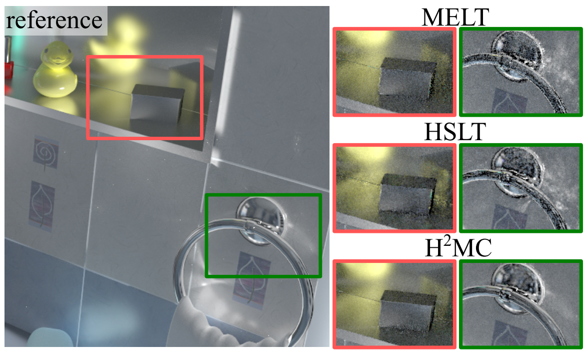


Fig. 21: Equal-time comparison (10 min) of differential MCMC methods. While Manifold exploration light transport (MELT, Sec 5.1) and Half-vector space light transport (HSLT, Sec 5.2) use geometry differentials to effectively explore some types of paths, Hessian-Hamiltonian Monte Carlo (H^{2}MC, Sec 5.3.1) adapts the mutations according to the target function derivatives and thus delivers better performance in this case. Image courtesy of Tzu-Mao Li.

6 Adaptive Markov chain Monte Carlo

Defining well-balanced mutations is one of the crucial factors for fast convergence of any MCMC algorithm. It is, however, almost impossible to design a set of mutations that is good for all possible scenes. Usually, the user of an MCMC algorithm must hand-tune some mutation parameters, for example the shape of the small step proposal distribution in PSSMLT (Sec. 4.1). This is however not always possible, since the optimal parameter can differ for various types of paths. To avoid hand tuning of these parameters, one can employ the so called *Adaptive Markov chain Monte Carlo* (AMCMC) [31]. AMCMC learns from the past proposals (both accepted or rejected) and automatically tunes the parameters of the mutations. In the following text, we first discuss theory and then describe an algorithm that introduced AMCMC to light transport.

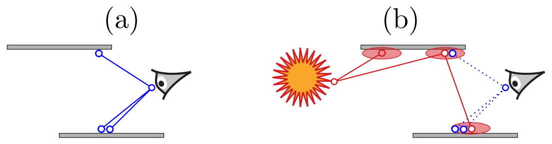


Fig. 22: Stochastic progressive photon mapping (SPPM) [35] works in two passes: (a) First, it generates a set of measurement points (blue) from the camera. (b) Second, a set of photon paths (red) from the light sources is generated and the image is computed by splatting photons (vertices of photon paths) onto nearby measurement points (red ellipse). This corresponds to performing density estimation over photons at each measurement point. The generation of measurement points and photon paths is iteratively repeated and the image estimates from each iteration are averaged. To ensure consistency of the method the density estimation radius is shrunk in each iteration.

6.1 General AMCMC

We give here a brief description of one of the AMCMC methods, *Controlled Markov chain Monte Carlo* [32]. For a more comprehensive overview of AMCMC, we refer readers to a survey by Andrieu and Thoms [33].

Unlike the classical MCMC, in the AMCMC algorithm the transition function t (Sec. 2.1) changes with each step. In controlled MCMC, this change is condensed into one parameter θ_i of an otherwise fixed transition function $t(\theta_i, u \to v)$. Given an initial value of the parameter θ_0, the value for i-th step θ_i is computed from the previous samples as

$$\theta_{i+1} = H(i, \theta_i, u_0, \dots u_i). \tag{23}$$

Here $u_0, \dots u_i$ are all samples generated up to the i-th step and the function H updates the parameter according to the history of samples and the current parameter value.

Introducing the adaptivity results in the chain no longer being Markovian. This means that the proof of convergence towards the stationary distribution no longer holds (and the algorithm may diverge [33]). To ensure convergence we have to impose a *diminishing adaptation* condition [33]

$$\lim_{i \to \infty} |H(i, \theta_i, u_0, \dots u_i) - \theta_{i+1}| = 0. \tag{24}$$

In other words, the adaptation must diminish over time.

6.2 Robust Adaptive Photon Tracing

The AMCMC was first introduced to light transport by Hachisuka and Jensen [34]. Their method improves the *Stochastic progressive photon mapping* (SPPM) algorithm [35], described in the caption of Fig. 22, by using MCMC to distribute photons into the scene.

6.2.1 Guiding photons in SPPM

The original SPPM suffers in scenes where only a small portion of photons contributes to the image. In the new method, MCMC is used to generate only photons that contribute to the image (Fig. 23). This is achieved by defining the so called *visibility* target function. The visibility function is a binary function, which is non-zero for photon paths where at least one photon contributes to the image. Such a target function is easily explored by MCMC (a contributing photon path is always accepted) and thus the chain quickly converges to the stationary distribution.

6.2.2 *Mutation adaptation*

The algorithm operates in the primary sample space and adapts the small step mutation (Sec. 4.1). The mutation is adapted using a single parameter $\theta \in [0, 1]$ that controls the width of the proposal distribution. For θ equal to zero, the proposal will always be equal to the current sample, while for θ equal to one the proposal will be a uniformly selected vector from the whole state space. θ is then adapted based on the acceptance rate A_i of the states proposed so far

$$\theta_{i+1} = \theta_i + \frac{1}{i}(A_i - A^*) \tag{25}$$

If the acceptance rate is below the optimum A^* (23.4% is considered optimal under certain conditions [36]), then θ will decrease and thus proposals will be closer to the current state and will likely have similar target function value (and will more likely be accepted). Otherwise, θ increases and therefore proposals will be farther from the current state.

6.2.3 *Discussion*

While the algorithm certainly outperforms the original SPPM, the photon distribution in the visible regions can be highly nonuniform and thus the resulting error distribution in the image may be far from uniform. This issue is addressed by some of the methods discussed later (Sec. 9.3). The algorithm efficiency could be further improved by adapting the mutations independently for different parts of the state space [37].

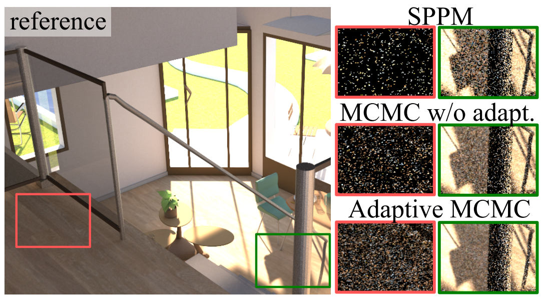


Fig. 23: The scene is lit from the outside and thus only few photons in Stochastic progressive photon mapping (SPPM) [35] reach the visible part. While using the method by Hachisuka and Jensen (Sec. 6.2) without adapting the mutations helps to alleviate this problem, the same method with adaptive MCMC is even more efficient. All algorithms ran for 10 minutes.

7 Better global exploration I: Tempering and Replica exchange

In this section we describe a more robust MCMC algorithm *Replica exchange* (RE) [38] which builds on top of Metropolis-Hastings (Sec. 2.2). First, we show how RE achieves better *global exploration* than the original Metropolis-Hastings. After that we discuss two light transport algorithms that use RE in different ways.

7.1 The Replica exchange algorithm in general

The Replica exchange algorithm (a.k.a. Parallel tempering) improves global exploration of the state space by running M chains in parallel (each is updated as in MH) that influence each other by a special step called *replica exchange*. This step exchanges the current state of one chain with the current state of another chain (Fig. 24). In order to be able to swap the states without changing the stationary distribution of these chains, the swap between a chain c_i and a chain c_j must be performed with the probability

$$s(i, j) = \frac{\pi_i(u_j)\pi_j(u_i)}{\pi_i(u_i)\pi_j(u_j)}, \tag{26}$$

where u_i is the current state of the chain c_i and π_i is its target function (similarly for the chain c_j).

The swapping makes the chains become dependent and they are no longer Markov. However, the whole process of M-chains forms one Markov chain that will converge to the distribution of $\Pi = \pi_1 \times \ldots \times \pi_M$. If the mutations ensure ergodicity, then each chain c_i will generate a distribution that converges to the desired distribution π_i^*.

7.2 Choosing target functions for Replica exchange

To enable better exploration of the state space, each chain should have a differently modified target function. The common strategy in general MCMC is to use function tempering [39]. One takes the original target function, which is hard to explore and then the function is flattened as follows

$$\pi_i = \pi^{1/T_i}, \tag{27}$$

where $T_i \geq 1$ is the *temperature* of the chain c_i. The higher the temperature, the flatter the target function and the more easily can the algorithm explore the state space (Fig. 25).

The chains are usually sorted according to their temperatures: $T_1 \leq T_2 \leq \ldots \leq T_M$ to the highest. The swaps are then commonly only performed between neighboring chains, so that the swap probability $s(i, j)$ is kept high. One also sets T_1 equal to one (i.e. first chain uses the original target function) in order to get good importance sampling of high contributing areas of the state space.

The number of chains and their temperatures are set so that the chain with the highest temperature is flat enough and so the swap probability is around 23.4% (which is considered to be ideal [40]). One can set the temperatures manually or use automatic methods [41].

7.3 Replica Exchange Light Transport

Replica exchange was first introduced into light transport in *Replica exchange light transport* (RELT) [42]. RELT uses several chains that operate in the primary sample space

$$\begin{tikzcd}
c_1 & \boxed{u_1} \arrow[r] & \boxed{u_2} \arrow[r] & \boxed{u_3} \arrow[dr] & \boxed{v_3} \arrow[r] & \boxed{v_4} \arrow[r] & \boxed{v_5} \\
c_2 & \boxed{v_1} \arrow[r] & \boxed{v_2} \arrow[r] & \boxed{v_3} \arrow[ur] & \boxed{u_3} \arrow[r] & \boxed{u_4} \arrow[r] & \boxed{u_5}
\end{tikzcd}$$

Fig. 24: Replica exchange (Sec. 7.1) between two chains c_1 and c_2. After several standard mutations their states u_3 and v_3 are exchanged. The chains then again continue mutating these states independently.

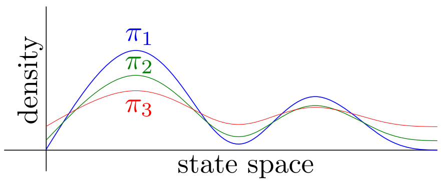


Fig. 25: The figure shows target functions with increasing temperature. The target function π_1 with the lowest temperature (blue) has the tallest peaks and the lowest valleys and thus is much harder to explore than the target function π_3 with the highest temperature (red).

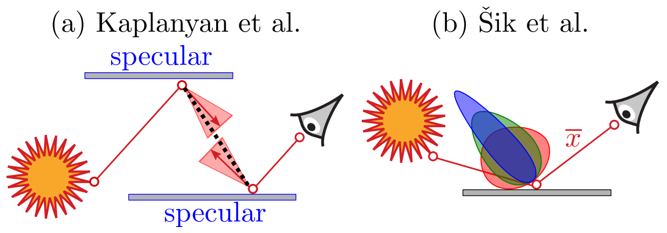


Fig. 26: (a) Kaplanyan et al. (Sec. 7.4) introduced regularization that allows connection of the two specular vertices. (b) Šik et al. (Sec. 7.5) propose increasing roughness of a BRDF (from the blue narrow peak to the red wide lobe) to improve exploration since paths like $\overline{x}$ are more likely to be accepted.

(Sec. 4.1) and differ only in which mappings they use to map samples to the path space. Each chain is therefore more effective at exploring different parts of the path space. As in any RE algorithm, the chains exchange their states (i.e. the current sample of one chain becomes the current sample of the neighboring chain and vise versa). The state exchange helps to explore local subspace with the chain that uses a mapping better suited for this subspace.[3] Therefore it behaves similarly as a change of mapping in MMLT (Sec. 4.2) with all its advantages and disadvantages.

Like MMLT, RELT reduces overhead compared to PSSMLT by not computing all possible mappings for each sample. Unfortunately, exchanging states may also severely change the resulting path and thus limits the acceptance probability of such an exchange. This problem could be alleviated by applying inverse mappings similarly as in MMLT (Sec. 4.3.3), but it would involve changing the exchange acceptance probability to account for the inverse mapping.

Unfortunately, some types of paths (such as reflected caustics) cannot be effectively sampled by any of the chains (no mapping handles them well) and thus the possibility of a chain getting stuck is not entirely eliminated.

7.4 Path space regularization

To alleviate the problems with the hard-to-sample paths, Kaplanyan et al. [43] introduced the concept of path space regularization. A path that was previously difficult or even impossible to sample due to presence of specular/highly-glossy interactions (or due to point light source, etc.), can be more easily sampled after one or two of the specular interactions is regularized as shown in Fig. 26. To ensure consistency, the regularization must be gradually diminished with a carefully chosen rate. Note that fast diminishing of regularization leads to under-sampling of difficult to sample paths, while the opposite causes oversampling of these paths. Kaplanyan et al. further show under which conditions will a MCMC-based algorithm converge in the presence of regularization.

7.5 Tempering of the path contribution

Šik et al. [44] borrowed the idea of regularization and combined it together with Replica exchange. More specifically, they modify the MMLT algorithm (Sec. 4.2) by introducing chains with increasingly flatter target functions. The flattening (tempering) is achieved by the regularization, in this case performed by increasing the roughness of bidirectional reflectance function (BRDF) (Fig. 26). The authors point out that increasing roughness of all surfaces changes the simulation result too much, which results in low probability of Replica exchange moves. Thus they only change the roughness of BRDFs at the vertices that are not importance sampled by the sampling technique corresponding to the current mapping to the path space. Unlike in RELT, using this approach flattens the target function for all types of paths including reflected caustics.

The authors introduced two new strategies for Replica exchange moves:

- **Equi-energy moves.** Swaps between chains are performed for any two chains whose current target function value is equally high. This strategy theoretically increases the probability of swapping [45].
- **Importance-sampled permutations.** Instead of a swap between two chains, permutation of all current states of all chains is used. The permutations are proposed based on the resulting target function value of all chains and are always accepted.

The authors show that the classical swapping of states between neighboring chains achieves similar results as the permutations, while Equi-energy moves are less effective. Fig. 27 shows a comparison of this method with the original MMLT.

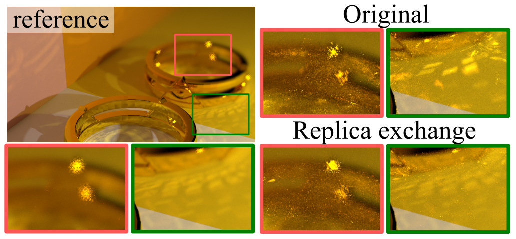


Fig. 27: Equal-time comparison of the original Multiplexed MLT (MMLT, Sec. 4.2) and the MMLT with Replica exchange (Sec. 7.5). The original MMLT under-samples some parts of the image, while over-sampling others due to poor global exploration. Replica exchange improves global exploration and thus its result has a more uniform error.

3. Imagine that one state randomly discovers a path which it can not effectively exploit due to its mapping, however it may exchange the state with a chain that uses better suited mapping and which can thus effectively sample the subspace around the discovered path.

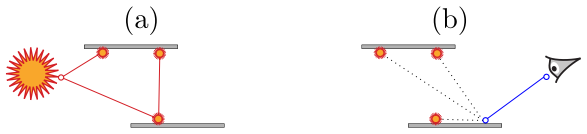


Fig. 28: Instant radiosity [47] computes the image in two steps: (a) Light paths (red) are traced from the light sources, at each vertex of these paths a virtual point light (VPL) is deposited. (b) Rays (blue) are shot from the camera, at each hit point with a scene the illumination is computed from every VPL which acts as a light source. The basic method only handles diffuse surfaces.

8 Better global exploration II: Multiple-try Metropolis

In this section we introduce the *Multiple-try Metropolis* (MTM) [46] algorithm that improves the distribution of MCMC samples by considering a whole set of proposals V mutated from the current sample (unlike standard Metropolis-Hastings that uses just one proposal). The final proposal is selected from the set of proposals V by importance sampling and thus has higher chance of acceptance compared to the one proposal used in Metropolis-Hastings. This in turn decreases the chance of MTM getting stuck in a local maximum. Note however that each iteration of MTM is quite expensive since not only a whole set of proposals V is generated, but a set of *competitors* U^* must be generated as well (see Sec. 4 of the supplemental material for more details). In the following text we describe two light transport algorithms that benefit from MTM despite its high computational cost.

8.1 Metropolis Instant Radiosity

MTM was introduced to light transport in an algorithm called *Metropolis instant radiosity* (MIR) [48]. MIR builds on light transport algorithm *Instant radiosity* [47], described in the caption of Fig. 28. The advantage of the algorithm is that a small set of virtual point lights (VPLs) is enough to be used for a quick rendering of the scene. To ensure that the VPLs capture illumination of a scene as closely as possible, they must be well distributed. This is were MIR steps in.

MIR combines Instant radiosity with the original MLT (Sec. 3.3), while replacing the standard Metropolis-Hastings algorithm with Multiple-try Metropolis. MCMC is only used here to distribute the VPLs. The target function is equal to path contribution as in MLT, so the VPLs are distributed according to how much energy they contribute to the image. Compared to other VPL-based methods, using MLT to distribute VPLs will result in VPLs being placed in areas that are more important for the current camera position. The disadvantage is however that once the camera moves, the VPLs should be generated again.

Since MCMC samples are usually heavily correlated, illuminating a scene from a small set of correlated VPLs would lead to some serious artifacts. The authors have therefore decided to trade efficiency of generating VPLs for their better distribution and applied the MTM algorithm. Generating the VPLs is usually much cheaper than accumulating their illumination. Thus using MTM is advantageous. To further improve the distribution of VPLs, only bidirectional mutations from MLT are used (so as to avoid the correlation caused by the local exploration mutations). Fig. 29 shows a comparison of VPL-based methods where VPLs are distributed using MTM and MH.

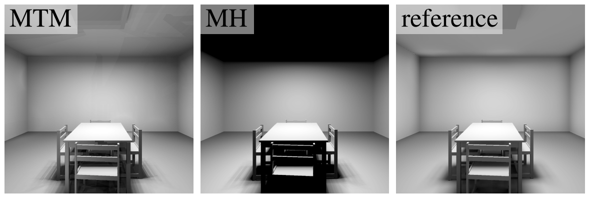


Fig. 29: Multiple-try Metropolis (MTM, Sec. 8) improves the distribution of 128 VPLs compared to Metropolis-Hastings (MH). The reference was computed using MH with 12800 VPLs. Image courtesy of B. Segovia.

8.2 Coherent MLT

Segovia et al. [49] point out that MTM can serve a different purpose than improving sample distribution. In their work MTM is utilized in the original MLT to generate coherent paths that are more easily raytraced by massive parallel hardware or wide instruction sets. As required by MTM, a set of proposed states V and a set of competitors U^* is generated. The states are generated using local exploration mutations (e.g. lens perturbation of MLT, Sec. 3.3) to ensure their coherency. While only one state is accepted, contribution from all the tentative states $V \cup U^*$ is accumulated using the expected value optimization (Sec. 2.6). Using MTM in this way exploits path coherency, but it also increases sample correlation.

9 Modifying target function for uniform image error

Usually the target function is proportional to the path contribution and thus brighter pixels receive more samples than the darker ones. While this approach decreases overall variance of the image estimate, the human eyes are similarly sensitive to relative variance in both bright and dark regions. Thus undersampling dark regions may result in longer rendering times, while the user waits till the dark regions have less visible noise. In this section we discuss methods that change the target function in order to reduce the overall error visible to the human visual system.

9.1 Two-stage and Multi-stage algorithm

Veach in his thesis [7, p. 357] points out that since the human eye is sensitive to contrast differences, each pixel should have the same relative error – in other words each pixel should receive approximately the same number of samples. To achieve this, he modifies the original MLT algorithm by running it in two stages. In the first stage, an image is computed with a low number of samples, from that we get the estimate for j-th pixel I'_j. In the second stage, a modified target function is used

$$\pi'(\overline{x}) = \begin{cases} \pi(\overline{x})/I'(\overline{x}) & \text{if } I'(\overline{x}) > 0 \\ \pi(\overline{x}) & \text{otherwise} \end{cases} \qquad (28)$$

where $I'(\overline{x})$ returns an estimate from the first stage for a pixel to which $\overline{x}$ contributes.

If the estimate from the first stage is accurate, in the second stage each pixel will have approximately the same number of samples. However, since the first stage was run with a low number of samples, in practice this will hardly be the case. The unconverged estimates I'_j may cause that some pixels will be heavily undersampled. Two-stage MLT may therefore converge more slowly than the original MLT [50].

9.1.1 Multi-stage and Noise-aware MLT

Hoberock et al. [50] improves on the two-stage MLT by proposing a *multi-stage* algorithm. Here in the first stage, the original MLT is executed on an image with drastically reduced resolution. Thus the image should converge quickly and I'_j estimates should be accurate. The following stage uses a slightly larger resolution and applies the modified target function Eq. 28 using up-sampled estimates from the previous stage. This way the algorithm continues through several stages until the last stage which uses the original resolution of the image. While the algorithm surpasses the two-stage version (Fig. 30), denoising the first stage without reducing the resolution could potentially work even better.

Furthermore, Hoberock et al. propose *Noise-aware* MLT, which estimates from the previous stage which pixels are more noisy according to a psychovisual method. While the samples are not distributed uniformly, the image should get noise-free more quickly when observed by a human.

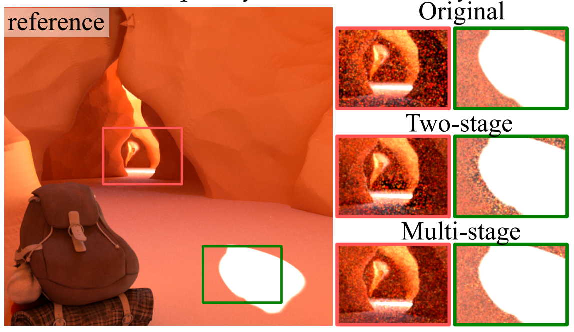


Fig. 30: Equal-time comparison (30 min) of renderings of a canyon lit through a narrow opening. The original MLT (Sec. 3.3) under-samples the dark regions. The two-stage method (Sec. 9.1) improves the sample distribution but fails to correctly modify the target function and thus often delivers worse result than the original. The multi-stage version (Sec. 9.1.1) modifies the target function more robustly and therefore achieves superior results.

9.2 Gradient domain MLT

The two/multi-stage algorithm adapts the target function based on a noisy image and thus can never be optimal. To avoid this issue, Lehtinen et al. [16] approach the uniform image error differently. They propose *Gradient domain MLT* (GDMLT) which builds on the original MLT with manifold exploration (MELT) (Sec. 5.1) and exploits the similarity in paths that contribute to neighboring pixels. More concretely, they use MCMC to compute both the original image I_O but also two difference images I_{dx}, I_{dy} that hold the information about finite difference of image contributions of paths that contribute to neighboring pixels. The resulting image is then computed from I_O, I_{dx}, I_{dy} via screened Poisson reconstruction (Fig. 31). In the case of L^2 Poisson reconstruction [51] GDMLT is an unbiased algorithm as proven by the authors.

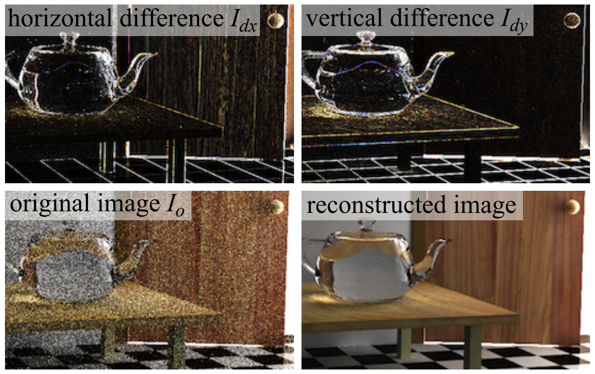


Fig. 31: Poisson reconstruction computes the final image in Gradient domain MLT (Sec. 9.2) from images with finite horizontal I_{dx} and vertical I_{dy} differences and from the original image I_O. Image courtesy of Lehtinen et al. [16]

.

9.2.1 Computing the difference images

To compute the difference images I_{dx}, I_{dy}, GDMLT uses MCMC to generate an original path $\overline{x}$ and a shifted path $\overline{x}_s$ together. The shifted path $\overline{x}_s$ is constructed from $\overline{x}$ as illustrated in Fig. 32. The same paths are used to compute all three images I_O, I_{dx}, I_{dy}. To ensure samples are distributed well in all images (i.e. in bright regions, but also in dark regions with high-frequency content), MCMC generates pairs of paths $\overline{x}$ and $\overline{x}_s$ using the following target function

$$\pi(\overline{x}, \overline{x}_s) = |f^*_d(\overline{x}, \overline{x}_s)| + \beta \frac{1}{4} f^*(\overline{x}), \tag{29}$$

where f^* is a scalar value of the path contribution and similarly f^*_d is a scalar difference of the contribution of $\overline{x}$ and $\overline{x}_s$. The user-defined factor β scales which component is deemed more important.

9.2.2 Comparison to MELT

Compared to MELT, the method usually needs fewer samples to get similar results (i.e. I_O can be quite noisy, while the reconstruction result will still be noise-free, see Fig. 31). On the other hand, GDMLT suffers more from poor global exploration caused by the higher variation of the target function (mainly coming from the scalar difference $|f^*_d(\overline{x}, \overline{x}_s)|$). The authors therefore propose to use L^1 reconstruction instead of L^2 to limit the occurrence of the overly-bright pixels caused by insufficient global exploration. While the use of L^1 reconstruction makes the method biased, it remains consistent.

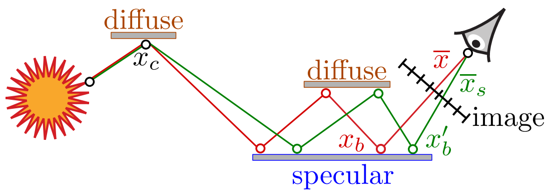


Fig. 32: Offset path sampling in Gradient domain MLT (Sec. 9.2). The shifted path $\overline{x}_s$ (green) is computed from the original path $\overline{x}$ (red) by first moving the first hit point from the camera x_b to x'_b so that $\overline{x}_s$ goes through a neighboring pixel. The algorithm then generates $\overline{x}_s$ vertices until it can connect to the original path at x_c.

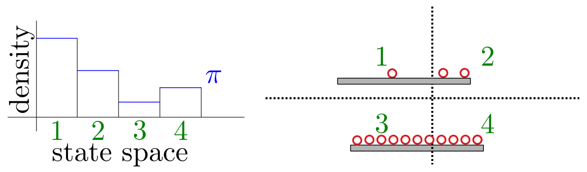


Fig. 33: According to Gruson et al. (Sec. 9.3.1), the desired target function for photon tracing is proportional to the inverse of the probability of receiving photon at a given point in the scene. This can be approximated by spatially dividing a scene and counting how many photons (red circles) fall in each bin (marked by green numbers).

9.2.3 Improved sampling for GDMLT

Besides other improvements of GDMLT, Manzi et al. [52] propose several orthogonal enhancements that reduce the variation of the target function. First, they propose two new path-shifting methods that are combined using multiple importance sampling. Second, to further increase the correlation between two neighboring paths, the shifted path is constructed from a pixel that is deemed to be closely correlated to the original path pixel not only by proximity but also by comparing multiple features (such as texture color, normals etc) that are computed for each pixel in a preprocess. The authors show that while these improvements lead to better global exploration, the method is still prone to artifacts due to the chain getting stuck.

9.3 Guiding photons for uniform image error

While the two/multi-stage and GDMLT algorithms improve sample placement where MCMC is used to generate whole paths, MCMC can be applied to just generate photons as in SPPM (Sec. 6.2).[4] To ensure that image converges uniformly, different approach must be taken for such methods.

Fan et al. [53] uses the vertices of paths generated by the original MLT as photons. While they are distributed mainly in important regions, their distribution is not ideal for photon mapping methods. Chen et al. [54] enhances the visibility target function used by Hachisuka and Jensen (Sec. 6.2) by incorporating the approximate density of photons contributing to each image pixel. The approximation uses an ad-hoc formula and is computed in a pre-pass from a limited number of samples and thus may not be optimal.

Zheng et al. [55] improves the previous method by utilizing visual importance [56] as a target function to guide the photons. Measurement points that are deemed to contribute less to the image by a pre-pass receive lower target function value and thus a light path that only contributes to them is less likely to be accepted. Gruson et al. [57] point out that while the other methods tend to distribute photons to important regions, the resulting error in the image is highly non uniform. We discuss their method in the following text.

9.3.1 Spatial Target Function

Gruson et al. [57] prove that an image has uniform relative error if each measurement point G has the same probability of receiving a non-zero contribution from any photon path

4. Since the density estimation radius is gradually shrinked in SPPM (Fig. 22), the target function of the MCMC algorithms based on SPPM changes during the rendering.

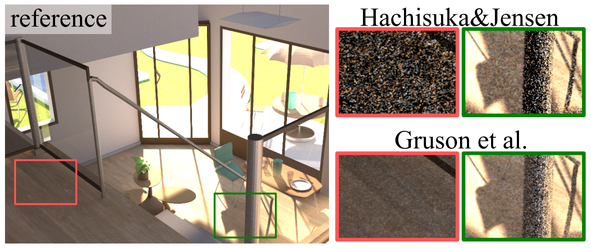


Fig. 34: Equal-time comparison of algorithms that improve photon guiding. While the method by Hachisuka and Jensen (Sec. 6.2) delivers photons to the visible regions, their distribution is highly non-uniform. The method by Gruson et al. (Sec. 9.3.1) generates photons in such a way that the resulting image has a more uniform error distribution.

(under simplifying assumptions, such as a diffuse BRDF). This is achieved if the target function value for photon path contributing to a measurement point G is equal to $1/P(G)$, where $P(G)$ is the probability of generating a photon path by uniform sampling of the primary sample space.

The authors use an approximation of the above target function based on statistics of previously generated photons (Fig. 33). This leads in practice to uniform image error, with the exception of pixels that display highly glossy materials. To improve the approximation over time, the method is split into iterations and each iteration uses target function based on statistics from all previous iterations.

Gruson et al. further propose several improvements to increase the efficiency of their method, namely they apply Replica exchange (Sec. 7). Fig. 34 shows a comparison with the method by Hachisuka and Jensen (Sec. 6.2).

10 Better global exploration III: Stratification

Stratified sampling is another approach to improving global exploration. Stratification is a Monte Carlo variance reduction technique that divides the sampled domain into subdomains where each receives the same number of samples and thus the whole domain is well-explored. In this section we discuss two methods that aim at improving the poor stratification of the usually highly correlated samples generated by MCMC.

10.1 Energy redistribution path tracing

Cline et al. [58] introduced *Energy redistribution path tracing* (ERPT) that aims at combining good stratification properties of regular Monte Carlo path tracing with efficient sampling of local high-contributing subspaces using the original local exploration mutations from MLT (Sec. 3.3). To do that they first generate a set of paths using a path tracer with a stratified sampler. Then the energy carried by each of these paths is redistributed to the surrounding pixels using MCMC sampling. In the following text we discuss the energy redistribution in more detail.

10.1.1 Energy redistribution

Using the rather non-standard ERPT notation and terminology, the energy of each path $\overline{x}$ is defined as $e(\overline{x}) = f(\overline{x})/p(\overline{x})$, where $f(\overline{x})$ is the contribution to the image

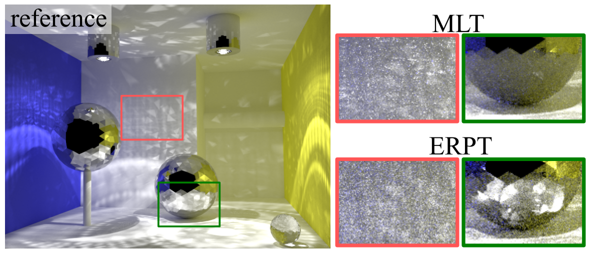


Fig. 35: Equal-time comparison of the original MLT (Sec. 3.3) and Energy redistribution path tracing (ERPT, Sec. 10.1). While MLT delivers less noisy image due to high sample correlation, ERPT discovers more important paths (like reflected caustics) thanks to its better global exploration. Scene courtesy of T. Hachisuka.

and $p(\overline{x})$ is the probability density with which $\overline{x}$ was sampled using the path tracer. To distribute this energy, $n_c(\overline{x})$ Markov chains are initialized with $\overline{x}$ and each of them is mutated using Metropolis-Hastings n_m-times. To mutate states, the authors use lens and caustic perturbations from MLT (Sec. 3.3), but other path space mutations can be used as well (e.g. manifold exploration, see Sec. 5.1). Each path generated by MCMC then deposits $e_d(\overline{x})$ amount of energy to the image

$$e_d(\overline{x}) = \frac{b^*}{n_c^* n_m}, \tag{30}$$

where b^* is the average energy of all initial paths and n_c^* is the average number of Markov chains per pixel. Note that b^* directly corresponds to the start-up weight (Sec. 2.5). In order to give unbiased results, each path $\overline{x}$ uses a different number of Markov chains

$$n_c(\overline{x}) = \lfloor \xi + \frac{e(\overline{x})}{n_m e_d(\overline{x})} \rfloor = \lfloor \xi + n_c^* \frac{e(\overline{x})}{b^*} \rfloor, \tag{31}$$

where $\xi \sim \mathrm{U}(0,1)$ is a uniformly distributed random number. Note that the number of chains for a given initial path is proportional to the path energy, thus more important initial paths may deposit their energy to more samples and thus decrease the chance of generating an overly-bright pixel.

10.1.2 Similarity to MLT

While ERPT may seem as very different from the original MLT, it is actually quite similar. In fact, given a set of initial path traced paths $\mathcal{X}$, using ERPT is equivalent to running $\sum_{\overline{x} \in \mathcal{X}} n_c(\overline{x})$ independent MLT algorithms without bidirectional mutations. Each $n_c(\overline{x})$ independent MLT algorithms are initialized with the same $\overline{x} \in \mathcal{X}$. The deposited energy $e_d(\overline{x})$ is equal to the common multiplier of samples in the original MLT (Eq. (16)).

However, due to the fact that ERPT uses many independent chains initialized from well stratified paths, the stratification is improved compared to the original MLT with a single chain. On the other hand, the number of mutations per each chain is decreased compared to the original MLT and thus the chains will more likely fail to converge to the desired distribution. This does not impair unbiasedness of ERPT, but it reduces its efficiency when dealing with hard-to-sample light paths. Fig. 35 shows a comparison between the original MLT and ERPT.

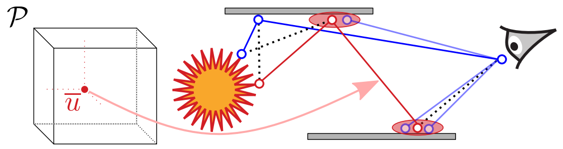


Fig. 36: In Metropolized bidirectional estimators (MBE, Sec. 10.2) a single light subpath (red) is created using a sample $\overline{u}$ from the primary sample space. This subpath is then connected (dashed black lines) to a randomly chosen camera subpath (opaque blue) using all mappings from bidirectional path tracing (not all of them are shown). Density estimation (red ellipse) is then performed at light subpath vertices using all vertices from camera subpaths.

10.2 Metropolised Bidirectional Estimators

The *Metropolised bidirectional estimators* (MBE) algorithm [59] improves upon Primary sample space MLT (PSSMLT, Sec. 4.1). First, it achieves better path stratification by generating camera subpaths using a path tracer with a stratified sampler, while using MCMC to generate only light subpaths. Second, Replica exchange is applied (Sec. 7) to improve global exploration. It uses two chains, one with the original target function (the same as in PSSMLT) and one with a binary visibility target function, where the visibility is defined to be one if the original target function is non-zero (similar to adaptive photon guiding, see Sec. 6.2.1). The authors show that using just two chains is usually sufficient (average swap probability is close to the ideal 23.4% [40]). Finally, MBE uses additional mappings to the path space and thus handles some hard-to-sample paths (e.g. reflected caustics) more efficiently.

10.2.1 Additional mappings

Compared to PSSMLT, which utilizes all mappings (sampling techniques) from bidirectional path tracing (Sec. 4.1.3), MBE uses additional mappings from *Vertex connection and merging* (VCM) [60] (a.k.a *Unified path sampling* [61]). These mappings correspond to density estimation with brute force path reuse, similarly to photon mapping (Fig. 22). To enable all VCM mappings, MBE is split into iterations, in each of them a set of stratified camera subpaths is generated first and then a given number of light subpaths is generated using MCMC sampling. Fig. 36 shows paths created using various VCM mappings given one light subpath.

10.2.2 Comparison to previous techniques

Due to the MBE features, the algorithm handles both stratification and difficult-to-sample paths. Fig. 37 shows comparison to ERPT and PSSMLT. The authors show that while their algorithm does not impose temporal coherence, animations generated by MBE do not suffer from the excessive temporal artifacts typical from most other MCMC approaches (see supplemental material of the original paper [59]). Furthermore, the authors point out that the MIS weights used in MCMC based sampling are not optimal and derive improved weights for the chain with visibility target function.

PART III: Discussion and conclusion

In this last part, we summarize and compare the described algorithms and offer possible directions for future work.

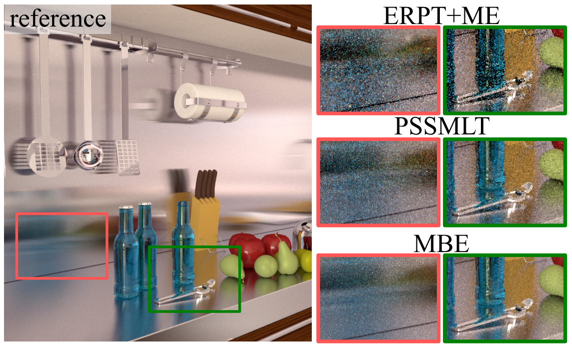


Fig. 37: Equal-time comparison (1 hour) of Energy redistribution path tracing (ERPT, Sec. 10.1) with manifold exploration (ME, Sec. 5.1), Primary sample space MLT (PSSMLT, Sec. 4.1) and Metropolized bidirectional estimators (MBE, Sec. 10.2). ERPT+ME enforces stratification by using many short Markov chains, however the chains fail to properly explore the complex glossy paths present in this scene. PSSMLT lacks stratification and density estimation techniques and thus delivers more noisy result than MBE.

11 Discussion

In Part II, we point out some important aspects of an MCMC algorithm: local exploration, global exploration, uniform image error. Here we compare how different algorithm categories achieve these goals and we offer possible directions for their combination and/or improvement.

11.1 Local exploration

Two categories of algorithms aim at improving local exploration. The algorithms discussed in Sec. 4 use a simplified state space called **primary sample space**. A simple mutation in this space is effective at sampling different types of paths due to many mappings to the path space. A different approach is taken by the algorithms introduced in Sec. 5, which adapt their mutations based on **differentials** of local geometry or of the target function. While these mutations are computationally expensive, their effectiveness usually outweighs the cost in the presence of difficult glossy/specular transport.

None of the mutations is the best under all circumstances. Even the simple mutation in the primary sample space can be the most efficient for some paths thanks to its low cost. For this reason, it would be best to choose the most appropriate mutation for each path automatically. While it is possible to combine mutations from the path space and the primary sample space (Sec. 4.3), it is still unknown how to choose the best mutation from all the available ones.

A different approach to improve local exploration is applied in Metropolized bidirectional estimators (Sec. 10.2), where density estimation along with brute force path reuse is used to efficiently handle paths that are otherwise difficult to explore. The authors show that their approach in many cases achieves better results than the complex mutations discussed above [59], but a more thorough analysis is needed to confirm their empirical results.

Since all the discussed mutations depend on user parameters, one can use **Adaptive Markov chain Monte Carlo** (Sec. 6) to automatically tune these. While it was shown that this improves the effectiveness of simple mutations in the primary sample space [34], this idea has never been used for more complex mutations. Furthermore, the existing methods adapt the mutations for the whole scene. Efficiency could be improved if the adaptation was localized [37].

11.2 Global Exploration

In this survey we discuss two general MCMC techniques that improve global exploration: **Replica exchange** (RE) (Sec. 7) and **Multiple-try Metropolis** (MTM) (Sec. 8). Both of them can be applied to improve performance of any of the previously discussed algorithms.

The advantage of Replica exchange is its little overhead compared to MTM. While it runs several Markov chains with different target functions, all their samples can be used for the quadrature and possibly weighted using MIS [59]. The main disadvantage of RE is that its performance highly depends on the number of chains and on their target functions. Tuning RE to run optimally is a difficult task.

On the other hand, Multiple-try Metropolis has only one parameter: the number of proposals generated per one sample. The disadvantage of MTM is its huge overhead with increasing number of proposals. While one can apply the expected value optimization to use all the proposals in the quadrature (Sec. 8.2), the effectiveness of this approach is reduced since the proposals are often heavily correlated (they are mutated from the same sample). For this reason MTM has so far been used mainly where the overhead is not an issue (VPL generation, see Sec. 8.1); it remains to be seen if the superior distribution of samples could outweigh the additional cost in other use-cases. Another interesting and yet to be explored option would combine MTM and replica exchange to further improve global exploration.

MCMC literature lists other general techniques for improving global exploration that have yet to be applied to light transport. One example is **Delayed rejection** [62], which allows to accept another proposal when the previous one was rejected (without repeating the current sample).

The main reason why the current MCMC algorithms may fail to fully explore the state space is their use of uninformed uniform mutations as a means of global exploration (the large step mutation used in the primary sample space is at least locally informed due to the inherent importance sampling, see Sec. 4.1). While these mutations potentially explore the whole state space, they are often rejected since they only rarely find an important path. Adapting these mutations using information either from previous samples (e.g. using adaptive MCMC) or from some pre-process [63], [64] could significantly improve their acceptance rate and thus effectiveness of global exploration.

11.3 Uniform image error and stratification

We discuss several methods in Sec. 9 that aim to achieve more uniform image error. Some (Sec. 9.3) are useful only for photon mapping-based methods (Fig. 22), however the Two/Multi-stage MLT or the Gradient domain MLT could be potentially combined with all the algorithms from the other categories. Since these algorithms modify the target function, a possible complication would arise only when combining them with Replica exchange which already uses modified target functions, though for a different purpose.

Achieving better global exploration by improving stratification of paths is discussed in Sec. 10. The Energy redistribution path tracing approach (see Sec. 10.1), using many short Markov chains initialized in different pixels, can be applied to any other algorithm. However this may actually hinder its efficiency, since the short chains will likely generate a distribution very different from the target distribution. The Metropolized bidirectional estimators algorithm (see Sec. 10.2) uses a more effective approach, since it generates truly stratified camera subpaths. The disadvantage is that it does not use MCMC for paths that are traced only from the camera (such as pure specular paths). It is also unclear how to apply such an approach to methods that mutate whole paths (i.e. the path space algorithms).

The discussed methods improve the stratification indirectly, but it would be beneficial if one could stratify directly the samples generated by a single Markov chain. Since the ability of MCMC to quickly explore important paths is due to the correlated samples it generates, one would have to find a balance between stratification and correlation.

11.4 Error analysis

Variance of the original MLT was analyzed by Ashikhmin et al. [65] who prove that the variance of a pixel is $\Theta(1/N)$ given N samples. However, their work uses two simplifying assumptions. First, variance is considered only for a single pixel, correlation among neighboring pixels is ignored. Second, the proof assumes that the Markov chain has already reached its stationary distribution. Since these assumptions are not true in reality, the analysis does not capture the true behavior of MLT or other MCMC algorithms. One possible approach to achieve more detailed and practical error estimation is the work of Kaplanyan et al. [43], who analyze the mixing rate of a Markov chain (i.e. how fast the chain reaches its stationary distribution).

11.5 Temporal stability

Temporal instability is perhaps the thorniest issue of current MCMC-based algorithms and it hinders their adoption in production renderers. While the authors of Metropolized bidirectional estimators (see Sec. 10.2) demonstrate the algorithm's superior temporal stability compared to other MCMC algorithms, this result is achieved only through better global exploration rather than by directly imposing temporal stability. Some instabilities may therefore still occur. Ensuring temporal stability comparable to ordinary path tracing remains an important topic for future work.

12 Conclusion and Open Problems

We have presented a survey of most of the MCMC-based light transport algorithms known to date, while discussing how each addresses various specific issues that emerge in conjunction with the use of MCMC in light transport simulation. While the existing methods are effective at some aspects, such as local exploration (e.g. thanks to considering geometry/target function differentials), numerous issues remain unresolved. Specifically, we identify three main problems that hinder a widespread adoption of MCMC-based algorithms in practice.

First, *global exploration* is an open issue. Scenes with many specular/glossy interactions separated by complex geometry or visibility yield numerous small islands of high-throughput in the path space. No current algorithm can, however, guarantee discovering all the relevant ones. While Replica exchange or Multiple-try Metropolis certainly improve global exploration, they do not solve the issue completely. One direction could involve using informed global mutations (e.g. utilizing path-guiding or adaptive MCMC).

Second, current MCMC algorithms suffer from severe *temporal instability*. The noise is often of low-frequency nature and as such, it is impossible to suppress by filtering. Currently the only somewhat effective approach to improve temporal stability is through improved global exploration, thereby limiting the appearing and disappearing of image features during an animation.

Finally, there is currently *no reliable error metric* to measure the algorithm progress and evaluate stopping criteria.

Despite the tremendous progress in MCMC light transport in the last several years, the above issues still prevent the use of MCMC-based algorithms in practice. This survey has identified numerous interesting directions for future research and it is our hope that these could eventually lead to the development of a universal light transport method based on MCMC.

Acknowledgments

The authors thank W. Jakob for his renderer Mitsuba, which was used to generate most of the images. We also thank M. Pharr, J. Hanika, A. S. Kaplanyan, T.-M. Li, B. Segovia, H. Otsu, T. Hachisuka, W. Jakob, and J. Lehtinen for permission to use their examples/scenes/images/source code. This work was supported by the grant SVV–2017–260452 and by the Czech Science Foundation grant 16–18964S.

References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, 1953.
[2] E. Veach and L. J. Guibas, "Metropolis light transport," in *SIGGRAPH '97*, 1997.
[3] W. Jakob, *Path Space Markov Chain Monte Carlo Methods in Computer Graphics*. Springer International Publishing, 2016.
[4] L. Tierney, "Markov chains for exploring posterior distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.
[5] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, 1970.
[6] S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.
[7] E. Veach, "Robust Monte Carlo methods for light transport simulation," Ph.D. dissertation, Stanford University, 1997.
[8] E. Nummelin, *General irreducible Markov chains and non-negative operators*. Cambridge University Press, 1984.
[9] L. Szirmay-Kalos, P. Dornbach, and W. Purgathofer, "On the startup bias problem of Metropolis sampling," Technical University of Budapest, Tech. Rep., 1999.
[10] E. Veach and L. Guibas, "Bidirectional estimators for light transport," in *Proc. Eurographics Rendering Workshop*, 1994.
[11] M. Schmidt, O. Lobachev, and M. Guthe, "Coherent Metropolis light transport on the GPU using speculative mutations," in *Journal of WSCG. 2016, vol. 24*, ser. Journal of WSCG. 2016, 2016.
[12] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 105–110, 03 1947.
[13] E. Veach and L. J. Guibas, "Optimally combining sampling techniques for Monte Carlo rendering," in *SIGGRAPH '95*, 1995.
[14] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering (Third Edition)*. Boston: Morgan Kaufmann, 2017.

[15] M. Pauly, T. Kollig, and A. Keller, "Metropolis light transport for participating media," in *Eurographics Rendering Workshop*, 2000.

[16] J. Lehtinen, T. Karras, S. Laine, M. Aittala, F. Durand, and T. Aila, "Gradient-domain Metropolis light transport," *ACM Trans. Graph. (SIGGRAPH 2013)*, vol. 32, no. 4, 2013.

[17] C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka, "A simple and robust mutation strategy for the Metropolis light transport algorithm," *Comp. Graph. Forum (Eurographics)*, vol. 21, no. 3, 2002.

[18] K. Zsolnai and L. Szirmay-Kalos, "Automatic parameter control for Metropolis light transport," in *EG 2013 Short Papers*, 2013.

[19] L. Szirmay-Kalos, B. Benedek, and M. Sbert, "Metropolis iteration for global illumination," *Journal of WSCG*, vol. 12, no. 1-3, 2004.

[20] T. Hachisuka, A. S. Kaplanyan, and C. Dachsbacher, "Multiplexed Metropolis light transport," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[21] H. Otsu, A. S. Kaplanyan, J. Hanika, C. Dachsbacher, and T. Hachisuka, "Fusing state spaces for Markov chain Monte Carlo rendering," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[22] J. Pantaleoni, "Charted Metropolis light transport," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 75:1–75:14, Jul. 2017.

[23] B. Bitterli, W. Jakob, J. Novák, and W. Jarosz, "Reversible jump Metropolis light transport using inverse mappings," *ACM Trans. Graph.*, vol. 37, no. 1, pp. 1:1–1:12, Oct. 2017.

[24] W. Jakob and S. Marschner, "Manifold exploration: A Markov chain Monte Carlo technique for rendering scenes with difficult specular transport," *ACM Trans. Graph.*, vol. 31, no. 4, 2012.

[25] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, 2007.

[26] A. S. Kaplanyan, J. Hanika, and C. Dachsbacher, "The natural-constraint representation of the path space for efficient light transport simulation," *ACM Transactions on Graphics*, vol. 33, no. 4, 2014.

[27] J. Hanika, A. Kaplanyan, and C. Dachsbacher, "Improved half vector space light transport," *Comp. Graph. Forum*, vol. 34, no. 4, 2015.

[28] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216 – 222, 1987.

[29] T.-M. Li, J. Lehtinen, R. Ramamoorthi, W. Jakob, and F. Durand, "Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics," *ACM Trans. Graph. (SIGGRAPH Asia 2015)*, vol. 34, no. 6, 2015.

[30] D. Piponi, "Automatic differentiation, C++ templates, and photogrammetry," *Journal of Graphics Tools*, vol. 9, no. 4, 2004.

[31] H. Haario, E. Saksman, and J. Tamminen, "An adaptive Metropolis algorithm," *Bernoulli*, vol. 7, no. 2, pp. 223–242, 04 2001.

[32] C. Andrieu and C. P. Robert, "Controlled MCMC for optimal sampling," Université Paris-Dauphine, Tech. Rep., 2001.

[33] C. Andrieu and J. Thoms, "A tutorial on adaptive MCMC," *Statistics and Computing*, vol. 18, no. 4, pp. 343–373, 2008.

[34] T. Hachisuka and H. W. Jensen, "Robust adaptive photon tracing using photon path visibility," *ACM Tr. Graph.*, vol. 30, no. 5, 2011.

[35] ——, "Stochastic progressive photon mapping," *ACM Trans. Graph. (SIGGRAPH Asia 2009)*, vol. 28, no. 5, 2009.

[36] J. S. Rosenthal, *Optimal Proposal Distributions and Adaptive MCMC*, 1st ed., ser. Handbooks of Modern Statistical Methods. Florida, USA: Chapman & Hall, CRC, 2011, ch. 4.

[37] R. V. Craiu, J. Rosenthal, and C. Yang, "Learn from thy neighbor: Parallel-chain and regional adaptive MCMC," *Journal of the American Statistical Association*, vol. 104, no. 488, pp. 1454–1466, 2009.

[38] R. H. Swendsen and J.-S. Wang, "Replica Monte Carlo simulation of spin-glasses," *Phys. Rev. Lett.*, vol. 57, pp. 2607–2609, 1986.

[39] D. J. Earl and M. W. Deem, "Parallel tempering: Theory, applications, and new perspectives," *Phys. Chem. Chem. Ph.*, vol. 7, 2005.

[40] Y. F. Atchadé, G. O. Roberts, and J. S. Rosenthal, "Towards optimal scaling of Metropolis-coupled Markov chain Monte Carlo," *Statistics and Computing*, vol. 21, no. 4, pp. 555–568, 2010.

[41] B. Miasojedow, E. Moulines, and M. Vihola, "An adaptive parallel tempering algorithm," *Journal of Computational and Graphical Statistics*, vol. 22, no. 3, pp. 649–664, 2013.

[42] S. Kitaoka, Y. Kitamura, and F. Kishino, "Replica exchange light transport," *Computer Graphics Forum*, vol. 28, no. 8, 2009.

[43] A. S. Kaplanyan and C. Dachsbacher, "Path space regularization for holistic and robust light transport," *Comput. Graph. Forum (Eurographics 2013)*, vol. 32, no. 2, pp. 63–72, 2013.

[44] M. Šik and J. Křivánek, "Improving global exploration of MCMC light transport simulation," in *ACM SIGGRAPH 2016 Posters*, 2016.

[45] M. Baragatti, A. Grimaud, and D. Pommeret, "Parallel tempering with equi-energy moves," *Stat. and Computing*, vol. 23, no. 3, 2012.

[46] J. S. Liu, F. Liang, and W. H. Wong, "The Multiple-Try method and local optimization in Metropolis sampling," *Journal of the American Statistical Association*, vol. 95, no. 449, pp. 121–134, 2000.

[47] A. Keller, "Instant Radiosity," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 1997.

[48] B. Segovia, J.-C. Iehl, and B. Péroche, "Metropolis instant radiosity," *Comput. Graph. Forum (Eurographics 2007)*, vol. 26, no. 3, 2007.

[49] B. Segovia, J.-C. Iehl, and B. Proche, "Coherent Metropolis light transport with Multiple-Try mutations," LIRIS UMR 5205 CNRS/INSA de Lyon, Tech. Rep. RR-LIRIS-2007-015, Apr. 2007.

[50] J. Hoberock and J. C. Hart, "Arbitrary importance functions for Metropolis light transport." *Com. Graph. Forum*, vol. 29, no. 6, 2010.

[51] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, Jul. 2003.

[52] M. Manzi, F. Rousselle, M. Kettunen, J. Lehtinen, and M. Zwicker, "Improved sampling for gradient-domain Metropolis light transport," *ACM Trans. Graph.*, vol. 33, no. 6, Nov. 2014.

[53] S. Fan, S. Chenney, and Y.-c. Lai, "Metropolis photon sampling with optional user guidance," in *Eurographics Symposium on Rendering (EGSR '05)*, 2005, pp. 127–138.

[54] J. Chen, B. Wang, and J.-H. Yong, "Improved stochastic progressive photon mapping with Metropolis sampling," *Comput. Graph. Forum (EGSR 2011)*, vol. 30, no. 4, 2011.

[55] Q. Zheng and C.-W. Zheng, "Visual importance-based adaptive photon tracing," *The Visual Computer*, vol. 31, no. 6, 2015.

[56] P. H. Christensen, "Adjoints and importance in rendering: An overview," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 3, 2003.

[57] A. Gruson, M. Ribardière, M. Šik, J. Vorba, R. Cozot, K. Bouatouch, and J. Křivánek, "A spatial target function for Metropolis photon tracing," *ACM Trans. Graph.*, vol. 36, no. 1, pp. 4:1–4:13, Nov. 2016.

[58] D. Cline, J. Talbot, and P. Egbert, "Energy redistribution path tracing," *ACM Trans. Graph. (SIGGRAPH 2005)*, vol. 24, no. 3, 2005.

[59] M. Šik, H. Otsu, T. Hachisuka, and J. Křivánek, "Robust light transport simulation via Metropolised bidirectional estimators," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 245:1–245:12, Nov. 2016.

[60] I. Georgiev, J. Křivánek, T. Davidovič, and P. Slusallek, "Light transport simulation with vertex connection and merging," *ACM Trans. Graph. (SIGGRAPH Asia '12)*, vol. 31, no. 6, 2012.

[61] T. Hachisuka, J. Pantaleoni, and H. W. Jensen, "A path space extension for robust light transport simulation," *ACM Trans. Graph. (SIGGRAPH Asia '12)*, vol. 31, no. 6, 2012.

[62] A. Mira, "On Metropolis-Hastings algorithms with delayed rejection," *Metron - International Journal of Statistics*, vol. 0, no. 3-4, 2001.

[63] H. W. Jensen, "Importance driven path tracing using the photon map," in *Eurographics Workshop Rendering*, 1995, pp. 326–335.

[64] J. Vorba, O. Karlík, M. Šik, T. Ritschel, and J. Křivánek, "On-line learning of parametric mixture models for light transport simulation," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[65] M. Ashikhmin, S. Premože, P. Shirley, and B. Smits, "A variance analysis of the Metropolis light transport algorithm," *Computers & Graphics*, vol. 25, no. 2, pp. 287 – 294, 2001.

Martin Šik is a Ph.D. candidate at the Charles University, Prague, supervised by Jaroslav Křivánek. His current research focuses on Markov chain Monte Carlo algorithms for robust light transport simulation. Besides his academic research, he works as a researcher and developer at Render Legion, a.s., where he helps to develop Corona Renderer. He received his Master's degree in computer graphics from Charles University, Prague in 2012.

Jaroslav Křivánek is an associate professor of Computer Science at Charles University, Prague, and Director of Research at Chaos Group. Jaroslav received his Ph.D. from INRIA Rennes and Masters degree from Czech Technical University, Prague. His primary research interest is in realistic rendering with the focus on Monte Carlo methods for light transport simulation.