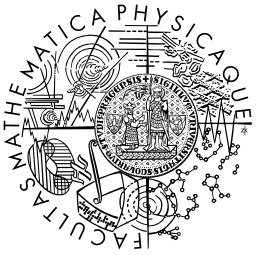

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

RNDr. Martin Šik

Global exploration in Markov chain Monte Carlo methods for light transport simulation

Department of Software and Computer Science Education

Supervisor of the doctoral thesis:	doc. Ing. Jaroslav Křivánek, Ph.D.
Study programme:	Informatics
Study branch:	Computer Graphics and Image Analysis

Prague 2018

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Global exploration in Markov chain Monte Carlo methods for light transport simulation

Author: RNDr. Martin Šik

Department: Department of Software and Computer Science Education

Supervisor: doc. Ing. Jaroslav Křivánek, Ph.D., Department of Software and Computer Science Education

Abstract: Monte Carlo light transport simulation has become a de-facto standard tool for photorealistic rendering. However, the algorithms used by the current rendering systems are often ineffective, especially in scenes featuring light transport due to multiple highly glossy or specular interactions and complex visibility between the camera and light sources. It is therefore desirable to adopt more robust algorithms in practice. Light transport algorithms based on Markov chain Monte Carlo (MCMC) are known to be effective at sampling many different kinds of light transport paths even in the presence of complex visibility. However, the current MCMC algorithms often over-sample some of the paths while under-sampling or completely missing other paths. We attribute this behavior to insufficient *global exploration* of path space which leads to their unpredictable convergence and causes the occurrence of image artifacts. This in turn prohibits adoption of MCMC algorithms in practice. In this thesis we therefore focus on improving global exploration in MCMC algorithms for light transport simulation. First, we present a new MCMC algorithm that utilizes replica exchange to improve global exploration. To maximize efficiency of replica exchange we introduce tempering of the path space, which allows easier discovery of important parts of the path space by the Markov chain. We further enhance replica exchange by designing novel replica exchange moves. Second, we present a different MCMC algorithm that is built upon the vertex connection and merging (VCM), a.k.a. unified path space sampling (UPS) algorithm. The path sampling techniques and subpath reuse from VCM/UPS leads to easier global exploration of the path space in the presence of glossy or specular transport. Besides the new algorithms that aim at improving global exploration, this thesis also contains a comprehensive survey of the Markov chain Monte Carlo methods used in light transport simulation.

Keywords: computer graphics, rendering, light transport simulation, Markov chain Monte Carlo, global exploration

I would like to express my sincere gratitude to my advisor, Jaroslav Křivánek, who dedicated lot of his time to guide me over the course of my doctoral studies and who taught me how to do a proper research and how to publish it. I also thank him for helping me with all the published work.

Further, I thank my colleagues Hisanari Otsu and Toshiya Hachisuka who have helped me with the research published in the thesis and I thank my colleagues Jiří Vorba and Jan Čermák for proofreading this thesis. I would also like to thank all my colleagues from the computer graphics group at the Faculty of Mathematics and Physics and the faculty staff for creating a great working environment.

Last but not least, I would like to thank my parents Jana and Martin who supported me and encouraged me throughout the years of my doctoral studies. I dedicate this work to my wife Markéta and to my children Lada and Martin, who always make me feel loved and happy.

This research has been supported by Charles University Grant Agency, projects GA UK no. 1362413 and no. 164815, by the grants SVV–2016–260332 and SVV–2017–260452, by the Czech Science Foundation grant 16-18964S.

Contents

Introduction **4**
1.1 Our approach . . . 5
1.2 List of original contributions . . . 6
1.3 Publications . . . 7

2 Introduction to light transport **8**
2.1 Light transport equation . . . 8
2.2 Path tracing . . . 9
2.3 Multiple importance sampling . . . 11
2.4 Light tracing . . . 11
2.5 Bidirectional path tracing . . . 11
2.6 Density estimation techniques . . . 12
2.7 Vertex connection merging/Unified path sampling . . . 12

Appendices **14**
2.A Computing path sampling technique pdfs . . . 14
2.A.1 Connection techniques . . . 14
2.A.2 Merging techniques . . . 15

3 Introduction to Markov chain Monte Carlo **16**
3.1 General MCMC algorithm . . . 16
3.2 Metropolis-Hastings algorithm . . . 17
3.3 A simple mutation example . . . 18
3.4 Convergence of the Metropolis-Hastings algorithm . . . 19
3.5 Using MCMC for quadrature . . . 20
3.6 Start-up bias . . . 21
3.7 Common optimization . . . 22

Appendices **23**
3.A Breaking the detailed balance condition . . . 23

4 MCMC algorithms for light transport simulation **24**
4.1 Metropolis light transport . . . 24
4.2 Simplified state space . . . 25
4.2.1 Primary sample space Metropolis light transport . . . 26
4.2.2 Multiplexed Metropolis light transport . . . 28
4.2.3 Fusing state spaces . . . 30
4.3 Better local exploration: Differential methods . . . 32
4.3.1 Manifold exploration light transport . . . 32
4.3.2 Half-vector space light transport . . . 34
4.3.3 Hamiltonian Monte Carlo . . . 37
4.4 Adaptive Markov chain Monte Carlo . . . 38
4.4.1 General AMCMC . . . 39
4.4.2 Robust adaptive photon tracing . . . 39
4.5 Better global exploration I: Tempering and replica exchange . . . 41
4.5.1 The replica exchange algorithm in general . . . 41

4.5.2 Choosing target functions for replica exchange 41
4.5.3 Replica exchange light transport 42
4.5.4 Path space regularization 43
4.6 Better global exploration II: Multiple-try Metropolis 43
4.6.1 Metropolis instant radiosity 44
4.6.2 Coherent MLT 45
4.7 Modifying target function for uniform image error 45
4.7.1 Two-stage and multi-stage algorithm 46
4.7.2 Gradient domain MLT 47
4.7.3 Guiding photons for uniform image error 49
4.8 Better global exploration III: Stratification 50
4.8.1 Energy redistribution path tracing 51
4.9 Discussion 52
4.9.1 Local exploration 52
4.9.2 Global Exploration 53
4.9.3 Uniform image error and stratification 54
4.9.4 Error analysis 54
4.9.5 Temporal stability 54

Appendices **55**
4.A Hamiltonian Monte Carlo 55
4.A.1 General Hamiltonian Monte Carlo 55
4.A.2 Approximating target function in Hessian-Hamiltonian Monte Carlo 56
4.B Multiple-try Metropolis 57

5 Improving global exploration by tempering and replica exchange **59**
5.1 Related work 61
5.2 Tempering of light transport simulation 61
5.2.1 Proposed tempering 62
5.3 Replica exchange moves and equi-energy sampler 68
5.3.1 Neighbor swapping 68
5.3.2 Importance-sampled permutations 68
5.3.3 Importance-sampled swaps 70
5.3.4 Equi-energy moves 71
5.3.5 Frequent equi-energy moves 72
5.3.6 Equi-energy sampler 72
5.3.7 Synthetic tests 74
5.3.8 Light transport tests 82
5.4 Implementation details 82
5.5 Results 86
5.5.1 The PSSMLT variant 87
5.5.2 The MMLT variant 90
5.6 Limitations 91
5.7 Future work 93
5.8 Conclusion 94

Appendices **95**
5.A Other tempering approaches . . . 95
5.A.1 Standard tempering . . . 95
5.A.2 Tempering by roughening all BSDFs . . . 97
5.A.3 Tempering of the connection BSDFs . . . 99
5.B Correctness of replica exchange moves . . . 99
5.B.1 Importance-sampled permutations . . . 100
5.B.2 Importance-sampled swaps . . . 100
5.B.3 Frequent equi-energy moves . . . 101
5.C Faster sampling of permutations . . . 101
5.C.1 Computing the normalization factor . . . 102
5.C.2 Importance sampling of permutations . . . 102

6 Robust light transport simulation via Metropolised bidirectional estimators **104**
6.1 Related work . . . 104
6.2 Algorithm design . . . 107
6.2.1 Requirements . . . 107
6.2.2 General algorithm design strategy . . . 107
6.2.3 Design choices . . . 108
6.3 Algorithm . . . 110
6.3.1 Light subpath contribution . . . 110
6.3.2 MCMC sampler details . . . 112
6.3.3 MCMC sampler implementation . . . 114
6.3.4 MIS weight computation . . . 114
6.4 Results . . . 116
6.5 Limitations and future work . . . 124
6.6 Conclusion . . . 126

Appendices **128**
6.A Algorithm components comparison . . . 128
6.B Mutating samples in the primary sample space . . . 129
6.C Our quest for a robust combination of MCMC and VCM/UPS . . . 129
6.C.1 Straightforward combination of PSSMLT and VCM/UPS . . . 129
6.C.2 Splitting generation of subpaths . . . 131
6.C.3 Final algorithm . . . 132

Conclusion **133**
7.1 Summary . . . 133
7.2 Future directions for improving global exploration . . . 134
7.3 Adopting Markov chain Monte Carlo in practice . . . 134

Bibliography **136**

List of Abbreviations **143**

Attachments **144**

Introduction

Monte Carlo light transport simulation has become a de-facto standard tool for photorealistic rendering. Recently, even the widely used rendering systems have adopted Monte Carlo algorithms for generating images [Christensen and Jarosz, 2016]. However, the algorithms used by the current rendering systems, e.g. path tracing [Kajiya, 1986] or in rare cases bidirectional algorithms [Lafortune and Willems, 1993, Veach and Guibas, 1994, Georgiev et al., 2012, Hachisuka et al., 2012], are often ineffective. Two major issues have been shown that compromise their efficiency: light transport due to multiple highly glossy or specular interactions, and scenes with complex visibility between the camera and light sources (Figure 1.1). While bidirectional algorithms can often handle light transport with multiple highly glossy or specular interactions, they fail in scenes with complex visibility [Popov et al., 2015]. Carefully avoiding such cases by modifying the scene can be cumbersome because to do that, the users need to understand how the algorithms work and when they may fail. Besides that, many applications require producing high quality imagery and tweaking scenes to satisfy imperfect algorithms is not acceptable.

To solve the above issues, Veach and Guibas [1997] have proposed Metropolis light transport (MLT), the first light transport algorithm based on Markov chain Monte Carlo (MCMC) [Metropolis et al., 1953]. Thanks to MCMC the Metropolis light transport algorithm can efficiently sample many different kinds of light transport paths even in the presence of complex visibility, however it often over-samples some of the paths while under-sampling or completely missing other paths. We attribute this behavior to insufficient *global exploration* in MCMC algorithms which leads to unpredictable convergence of the algorithm and causes the occurrence of many image artifacts (see Figure 1.2).

Since Metropolis light transport was developed, MCMC based light transport algorithms have received lots of attention. However, most of the recent works [Jakob and Marschner, 2012, Kaplanyan et al., 2014, Hachisuka et al., 2014, Li et al., 2015] focus on improving *local exploration*, i.e. effective sampling

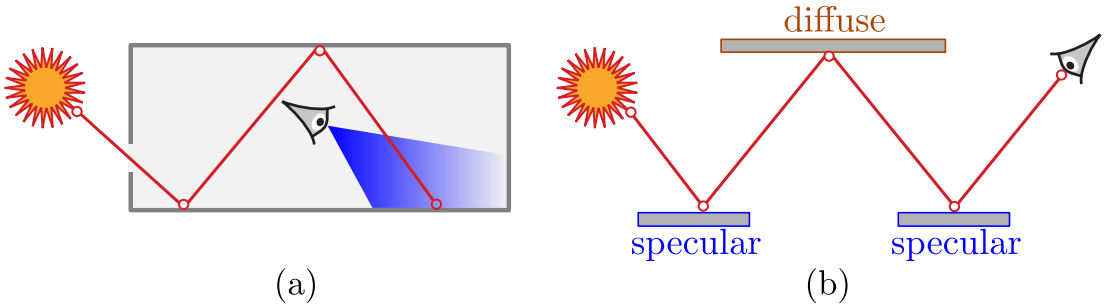

Figure 1.1: Two major issues that compromise efficiency of current light transport algorithms: (a) *Complex visibility* – the visible area (blue) is lit only through multiple reflections and/or much of the light emitted from the sources never reach the visible area. (b) *Complex glossy and specular transport* – light paths with chains of interactions on highly glossy (or specular) surfaces (e.g. a specular-diffuse-specular path shown in this figure).

of various light transport paths once a similar path was already discovered by the MCMC sampler. Meanwhile, the issue of insufficient global exploration is mostly ignored. Even the latest MCMC based algorithms still exhibit unpredictable convergence [Křivánek et al., 2014] and their results are not amenable to denoising [Zwicker et al., 2015]. This issue hinders application of these algorithms in practice, especially when rendering animations, where each frame exhibits different artifacts.

In this thesis we therefore focus on improving *global exploration* in MCMC algorithms. Our goal is to develop new MCMC algorithms that can discover and sufficiently sample all important paths, thus exhibiting a predictable convergence without producing image artifacts. We believe that achieving this goal is a fundamental prerequisite to the adoption of the MCMC algorithms in practice. This would in turn allow the users to render more complex scenes with difficult light transport.

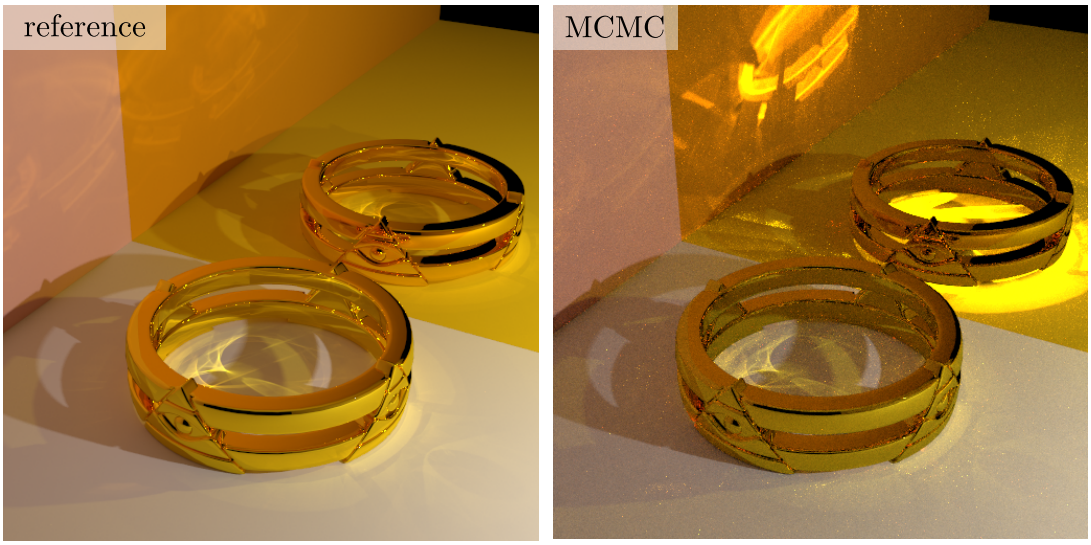

Figure 1.2: An image rendered by an example Markov chain Monte Carlo algorithm (manifold exploration light transport [Jakob and Marschner, 2012]) contains overly-bright regions of pixels. They are caused by oversampling some of the transport paths while failing to discover other important paths as a result of *insufficient global exploration*.

1.1 Our approach

In this dissertation we propose two different solutions for improving the so far insufficient global exploration in the existing Markov chain Monte Carlo methods for light transport simulation.

In Chapter 5 we present a new MCMC algorithm that utilizes *replica exchange* [Swendsen and Wang, 1986] to improve global exploration. To enable replica exchange, our algorithm runs several parallel Markov chains, where each generates paths from increasingly more *tempered* path space. We define the tempering of the path space so as to ensure that the Markov chain discovers all important parts of the path space and is able to frequently visit them. We further enhance replica exchange by designing new replica exchange *moves* that exchange the states of

the chains. The proposed replica exchange moves are general and can be used outside of light transport simulation. Finally, we introduce to light transport simulation the *equi-energy* sampler [Kou et al., 2006] that attempts to further improve replica exchange efficiency by allowing replica exchange moves between the current state of the chain and past states of another chain.

In Chapter 6 we propose a different MCMC based light transport algorithm that takes advantage of bidirectional estimators and subpath reuse to allow easier global exploration of the path space in the presence of glossy or specular transport. To be concrete, our MCMC algorithm is built upon the vertex connection and merging (VCM) [Georgiev et al., 2012], a.k.a. unified path space sampling (UPS) [Hachisuka et al., 2012] algorithm (referred to as VCM/UPS from now on). To avoid inheriting the issues of correlation and unpredictable convergence of MCMC, we apply replica exchange as in the previous algorithm. However, here replica exchange is just one of the algorithm's cornerstones and thus it is sufficient to use a mixture of only two target functions: one based solely on path visibility and the other on the path contribution. Furthermore, we reduce correlation in the image space and improve image-plane stratification by decoupling the sampling of light subpaths (i.e. subpaths from a light source) and camera subpaths (i.e. subpaths from the camera), and by sampling the camera subpaths by ordinary (independent) Monte Carlo.

1.2 List of original contributions

Our main contributions are two new light transport simulation algorithms based on Markov chain Monte Carlo that improve global exploration compared to the state-of-the-art. The algorithm, described in Chapter 6, exhibits in the tested scenes such a predictable convergence that it can be utilized in both fast preview of the rendered scene and also for rendering animations, unlike the previous MCMC based methods. Besides that, this thesis also includes a survey of the current light transport simulation algorithms based on Markov chain Monte Carlo. Below, we list our contributions in more detail.

Survey of Markov chain Monte Carlo methods in light transport simulation. Veach and Guibas were the first to use MCMC to solve light transport with their Metropolis light transport algorithm [Veach and Guibas, 1997] 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, in Chapter 4, we offer a comprehensive survey of MCMC methods used in light transport simulation. 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 improved in the near future.

Tempering of the path space. Kaplanyan and Dachsbacher [2013] introduced regularization/tempering of the path space to allow sampling of previously impossible or hard-to-sample paths. We show that their idea can be utilized to temper light transport simulation to allow for efficient replica exchange. Unlike in the original work, where the strength of regularization/tempering is limited

to avoid too much bias, we can use a more aggressive tempering due to the replica exchange setting, and thus we achieve more efficient exploration of the state space. Besides this approach we discuss other possible approaches to path space tempering and show why they are not as efficient.

New replica exchange moves. We have designed new replica exchange moves that result in a more efficient replica exchange algorithm. These moves are general and can be used outside of light transport simulation context. More specifically, we introduce here two novel moves: *importance-sampled permutations* and *importance-sampled swaps*. We have also improved the efficiency of the *equi-energy moves* algorithm by Baragatti et al. [2012] by allowing it to perform replica exchange moves more frequently.

Equi-energy sampling. To our knowledge we are the first to introduce the *equi-energy sampler* [Kou et al., 2006] to light transport simulation. Equi-energy sampling attempts to improve efficiency of replica exchange by allowing replica exchange moves between the current state of a Markov chain and past states of another Markov chain. In this thesis we compare its performance to the standard replica exchange in both light transport simulation and synthetic tests.

Robust combination of VCM/UPS and MCMC. Our algorithm introduced in Chapter 6 is the first Markov chain Monte Carlo algorithm that utilizes all path sampling techniques from VCM/UPS, including density estimation with subpath reuse. Robust combination of MCMC and VCM/UPS is not straightforward, and we therefore thoroughly discuss how to achieve it.

Identification of suboptimal multiple importance sampling weights. To our knowledge we are the first to point out that the multiple importance sampling weights used in some of the existing MCMC algorithms are in fact suboptimal. We show how to compute optimal weights according to the balance heuristic [Veach and Guibas, 1995] in our algorithm introduced in Chapter 6 and we demonstrate how using these MIS weights increases our algorithm's efficiency.

1.3 Publications

The content of this dissertation is based on the following publications:

- Šik, Martin and Křivánek, Jaroslav. Improving Global Exploration of MCMC Light Transport Simulation. *ACM SIGGRAPH 2016 Posters*, 2016.
- Šik, Martin and Otsu, Hisanari and Hachisuka, Toshiya and Křivánek, Jaroslav. Robust Light Transport Simulation via Metropolised Bidirectional Estimators. *ACM Transactions on Graphics (SIGGRAPH ASIA 2016)*, 35(6), November 2016.
- Šik, Martin and Křivánek, Jaroslav. Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation. To appear in *IEEE Transactions on Visualization and Computer Graphics*, 2019 (accepted in November 2018).

Chapter 5 is a significantly extended version of the research published in the first paper. The research presented in the second paper is the subject of Chapter 6. Chapter 4 contains the survey of Markov chain Monte Carlo methods published in the third paper.

2. Introduction to light transport

In this chapter we give a brief introduction to light transport simulation. For a more complete review, please refer to other sources [Veach, 1997, Pharr et al., 2017]. We first introduce the light transport equation in Section 2.1 and show how it can be computed using general Monte Carlo. We then briefly describe different algorithms that use ordinary Monte Carlo to solve the equation. These algorithms form the basis for some of the more advanced methods described later in the text.

2.1 Light transport equation

Light transport can be expressed by the path integral

$$I_j = \int_\Omega h_j(\overline{x}) f(\overline{x}) \mathrm{d}\mu(\overline{x}), \tag{2.1}$$

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 (Figure 2.1). A path of length k can therefore be represented as a vector of $k+1$ vertices $\overline{x} = (x_0, \ldots, x_k)$. h_j is the pixel filter of the pixel j, μ is the measure associated with the path space Ω [Veach, 1997], and the path contribution function f gives the amount of light energy transported along the path $\overline{x}$.

The path contribution function. The path contribution function f is defined as a product of several terms

$$\begin{aligned} f(x) = & L_e(x_0 \to x_1) \left(\prod_{i=1}^{k-1} G(x_{i-1} \leftrightarrow x_i) f_s(x_{i-1} \to x_i \to x_{i+1}) \right) \\ & G(x_{k-1} \leftrightarrow x_k) W_e(x_{k-1} \to x_k), \end{aligned} \tag{2.2}$$

where L_e is emission from the light source, f_s is the bidirectional scattering distribution function (BSDF), W_e is importance of the camera sensor and the geometric

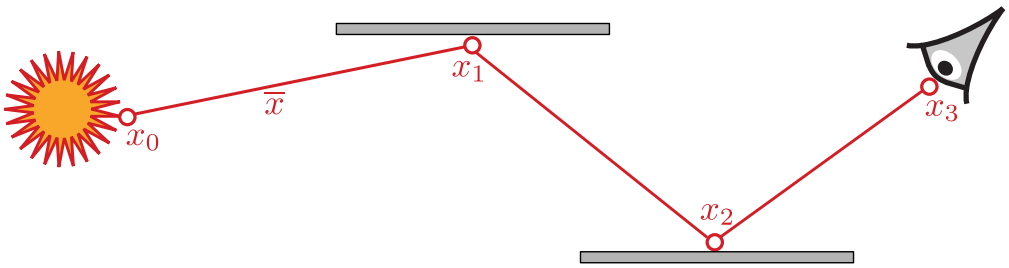

Figure 2.1: A light transport path $\overline{x}$ can be imagined as a polyline between a light source and the camera. The interior vertices correspond to light interactions with scene surfaces.

factor G is defined as

$$G(x_i \leftrightarrow x_{i+1}) = \frac{|\cos\theta_i^{\leftarrow}||\cos\theta_{i+1}^{\rightarrow}|}{\|x_i - x_{i+1}\|^2} V(x_i \leftrightarrow x_{i+1}). \tag{2.3}$$

Here $\theta_i^{\leftarrow}$ is the angle between the surface normal at x_i and direction from x_i towards x_{i+1}, while $\theta_{i+1}^{\rightarrow}$ is the angle between the surface normal at x_{i+1} and the opposite direction. V is a binary visibility function indicating whether there is no obstacle between the given vertices.

Monte Carlo estimator. Given a *path sampling technique* (e.g. path tracing, discussed next) 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

$$\langle I_j \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\overline{x}_i) f(\overline{x}_i)}{p(\overline{x}_i)}. \tag{2.4}$$

Note that a different estimate is computed for each pixel of the image (indexed by j). If the $p(\overline{x}_i) > 0$ whenever $h_j(\overline{x}_i) f(\overline{x}_i) > 0$, it can be proven that the above estimator is unbiased, i.e. $\mathbb{E}[\langle I_j \rangle] = I_j$. The variance of this estimator will depend on the number of samples N and on how similar the pdf p is to the integrand $h_j \cdot f$.

2.2 Path tracing

The standard path sampling technique for computing the image estimate using Equation (2.4) is the so called *unidirectional path tracing* [Kajiya, 1986]. Pseudocode of unidirectional path tracing is shown in Figure 2.2. The algorithm generates an estimate of each pixel by generating a number of paths (samples). Each path starts at the camera by tracing a ray through the estimated pixel (line 2). When the ray hits the surface (line 3), we either terminate the path and accumulate its contribution if a light source was hit (line 4) or we generate a new direction on the hit surface and continue tracing the path (line 7). If the ray leaves the scene (i.e. does not hit any surface), the path is terminated. Note that in a practical implementation, a path is usually also terminated if its potential contribution is deemed to be very low (to ensure unbiasedness after such termination, one should apply Russian roulette [Veach, 1997]).

```
1: for i = 0 to the number of paths/samples do
2:     Trace a ray through j-th pixel
3:     while Ray hits a surface do
4:         if Ray hits a surface of a light source then
5:             Accumulate contribution
6:         else
7:             Trace a ray in a new direction from the hit surface
```

Figure 2.2: Estimation of a single pixel j using unidirectional path tracing.

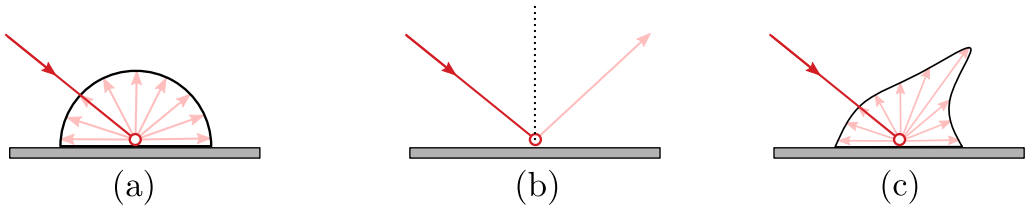

Figure 2.3: Different types of bidirectional scattering distribution functions (BSDF): (a) An ideal diffuse BSDF that reflects the light in all directions independently on the incoming direction. (b) Specular BSDF that reflects the incoming light according to the law of reflection. (c) A more realistic BSDF that combines a diffuse-like wide lobe with a glossy peak around the ideal reflection direction.

In order to reduce the variance of the estimator (Equation (2.4)), one should generate paths from a distribution with probability density proportional to the path contribution. While this is very hard to achieve in practice, one can usually lower the variance by *local importance sampling*. In unidirectional path tracing this translates to sampling a new direction after hitting the surface proportionally to the surface bidirectional scattering distribution function (BSDF). This BSDF defines how light reflects from a given surface, see Figure 2.3 for examples of different BSDFs.

To further improve the path tracing algorithm, *next event estimation* can be added. Instead of randomly hitting a light source, a position on a light source can be randomly selected and then deterministically connected to any vertex (surface hit) of the generated path. Figure 2.4 demonstrates the difference between path tracing with next event estimation and unidirectional path tracing. Ideally, one would like to use both techniques (randomly hitting a light source is efficient for a path with specular interactions, while next event estimation is more effective at sampling a diffuse surface lit by a small light source). The contribution of paths generated by both techniques can be effectively combined as discussed below.

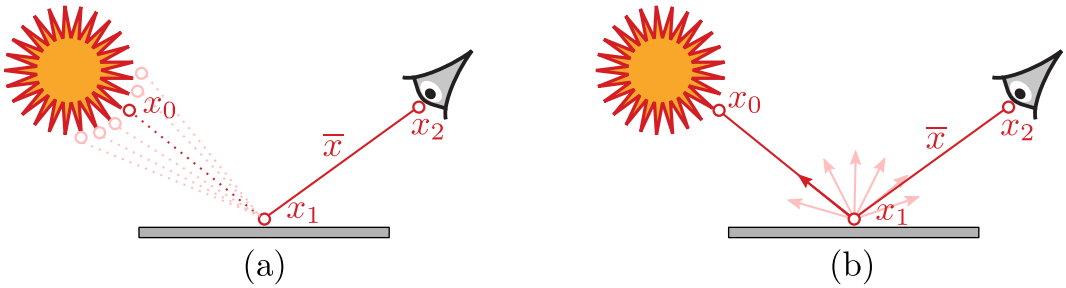

Figure 2.4: We can construct the same light path $\overline{x}$ in various ways. Given already sampled vertices x_1 and x_2, the vertex x_0 can be sampled by (a) selecting x_0 on a light source (next event estimation) or (b) sampling direction from x_1 and intersecting the light by a corresponding ray (unidirectional path tracing).

2.3 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. (2.4). 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 (e.g. path tracing with next event estimation, unidirectional path tracing) differ in the way they sample different vertices of the path (Figure 2.4). We can then reduce the variance if we combine path sampling techniques into one weighted sum

$$\langle I_j \rangle = \sum_{m \in \mathcal{M}} \frac{1}{N_m} \sum_{i=1}^{N_m} w_m(\overline{x}_{m,i}) \frac{h_j(\overline{x}_{m,i}) f(\overline{x}_{m,i})}{p_m(\overline{x}_{m,i})}. \tag{2.5}$$

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 [Veach and Guibas, 1995]

$$w_m(\overline{x}) = \frac{N_m p_m(\overline{x})}{\sum_{m' \in \mathcal{M}} N_{m'} p_{m'}(\overline{x})}. \tag{2.6}$$

2.4 Light tracing

Unlike in path tracing, where paths start by tracing a ray from the camera, in *light tracing* [Dutré and Willems, 1995] one starts a path by randomly selecting a position on a light source and tracing a ray in a random direction. The path then bounces off surfaces as in path tracing until it reaches a camera sensor or is deterministically connected to it (similarly to next event estimation in path tracing). The advantage of light tracing compared to path tracing is that it can generate some type of paths more efficiently (e.g. caustics, light focused by a mirror or a glass hitting a directly visible diffuse surface), however other paths are handled poorly by light tracing (directly seen mirror reflection of a light source).

2.5 Bidirectional path tracing

One can combine path tracing and light tracing in one *bidirectional path tracing* (BDPT) algorithm [Lafortune and Willems, 1993, Veach and Guibas, 1994], which can effectively sample many different types of paths. The algorithm generates a *camera subpath* from the camera as in path tracing and a *light subpath* from the light source as in light tracing and then it combines these subpaths into full paths by deterministically connecting their vertices (see Figure 2.5). The contribution of all the paths is then used for image estimation. To efficiently combine contributions from all the path sampling techniques (each technique corresponds to given length of a camera subpath and a given length of a light

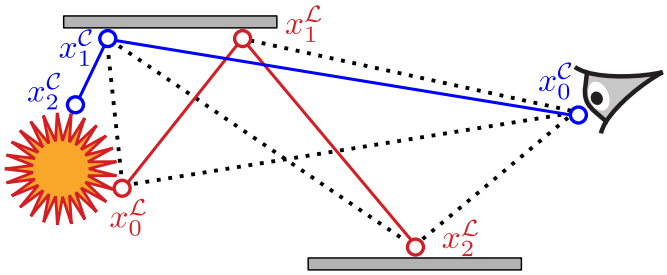

Figure 2.5: Bidirectional path tracing (Section 2.5) creates full paths by connecting (dashed line) a vertex of a camera subpath $x_i^{\mathcal{C}}$ (blue) with a vertex of a light subpath $x_j^{\mathcal{L}}$ (red). For example one path is created by connecting $x_1^{\mathcal{C}}$ with $x_2^{\mathcal{L}}$ yielding a path $\overline{x}$ with vertices $x_0^{\mathcal{L}}, x_1^{\mathcal{L}}, x_2^{\mathcal{L}}, x_1^{\mathcal{C}}, x_0^{\mathcal{C}}$. Note that each such combination of subpaths corresponds to one path sampling technique with a properly defined probability density function value.

subpath, see Appendix 2.A), one can apply multiple importance sampling (see Section 2.3).

2.6 Density estimation techniques

While bidirectional path tracing can sample many types of paths, paths that contain specular-diffuse-specular interactions (e.g. reflected caustics) cannot be effectively sampled by it. One way to sample such paths is to apply density estimation with brute force path reuse. This idea was first presented by Jensen [1996] in his *photon mapping* algorithm. Photon mapping first generates a batch of photon paths from light sources and then estimates the pixel values by performing density estimation over the vertices (photons) of these paths. The algorithm was later improved by Hachisuka and Jensen with their *progressive photon mapping* [2008] and *stochastic progressive photon mapping* (SPPM) [2009]. Figure 2.6 describes the SPPM algorithm in more detail.

2.7 Vertex connection merging/Unified path sampling

Stochastic progressive photon mapping was later combined with bidirectional path tracing into one robust algorithm *vertex connection merging/unified path sampling* (VCM/UPS) [Georgiev et al., 2012, Hachisuka et al., 2012]. This algorithm can create paths by either connecting a light subpath and a camera subpath (as in bidirectional path tracing) or by merging them, which is equivalent to performing density estimation at a given camera subpath vertex over nearby light subpath vertices (as in stochastic progressive photon mapping). See Appendix 2.A for more information about the different path sampling techniques. Note that to enable the merging, one must either generate a large batch of light subpaths before generating a camera subpath, or vise versa (when performing

Figure 2.6: Stochastic progressive photon mapping (SPPM) [Hachisuka and Jensen, 2009] 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 by a given amount.

density estimation over camera subpath vertices at a given light subpath vertex). While this algorithm can effectively sample most types of paths, it fails when rendering a scene where light has to bounce several times before reaching the camera, due to low probability of successful connection or merging [Popov et al., 2015]. In the following text, we will describe Markov chain Monte Carlo which allows for light transport algorithms that usually handle such scenes more effectively, while having a potential to be very effective at generating many different types of paths.

Appendix

2.A Computing path sampling technique pdfs

In this section we give a brief overview on how to compute the pdfs of various path sampling techniques used in vertex connection merging/unified path sampling (VCM/UPS) [Georgiev et al., 2012, Hachisuka et al., 2012]. We start this short overview by dividing the path sampling techniques into two groups: *connection techniques* and *merging techniques*. While the connection techniques were already used in bidirectional path tracing, merging techniques are specific to VCM/UPS. In the following text we describe how to compute the pdfs of both of these techniques.

2.A.1 Connection techniques

The connection techniques create a full path by connecting two vertices of a light subpath (subpath that starts on a light source) and a camera subpath (subpath that starts at the camera). Different connection techniques then vary in the length of the connected light subpath and camera subpath (including degenerated subpaths consisting of a single or zero vertices). In the following text we will note (s, t) connection technique as the technique that connects a light subpath of s vertices with a camera subpath of t vertices. Figure 2.7 shows examples of different connection techniques.

Since the connection is deterministic, the probability $p_C^{(s,t)}$ of generating a given full path $\overline{x} = x_0^{\mathcal{L}}, \ldots, x_s^{\mathcal{L}}, x_t^{\mathcal{C}}, \ldots, x_0^{\mathcal{C}}$ using the (s, t) connection technique is a product of pdfs of sampling each subpath

$$p_C^{(s,t)}(\overline{x}) = p(x_0^{\mathcal{L}}, \ldots, x_s^{\mathcal{L}}) p(x_0^{\mathcal{C}}, \ldots, x_t^{\mathcal{C}}). \tag{2.7}$$

Probability of sampling a given subpath (light or camera subpath) is then computed as a product of probabilities of sampling each of its vertices given a pre-

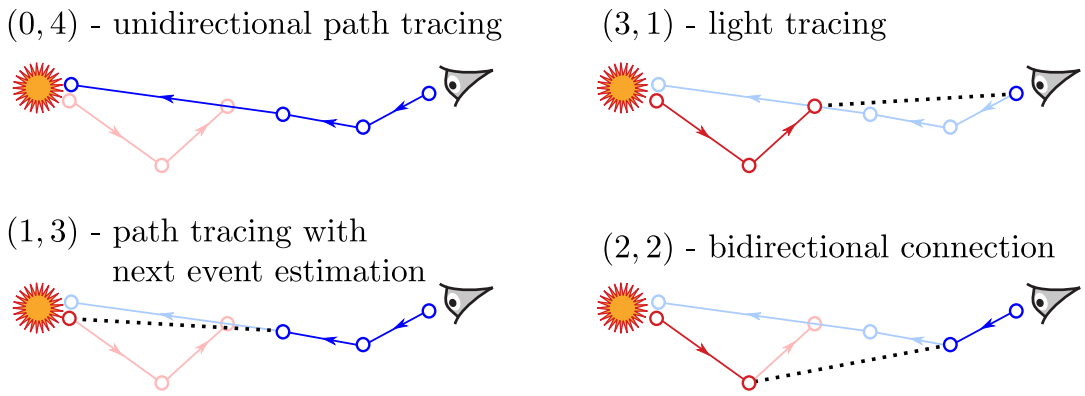

Figure 2.7: Examples of different connection techniques using the same light (red) and camera (blue) subpath to construct different full paths. The connection between the last camera vertex and light vertex is shown as a black dashed line. For each technique we list in the parentheses the number of light and camera subpath vertices they use.

ceding vertex

$$p(x_0, \ldots, x_n) = p(x_0)p(x_0 \to x_1) \cdots p(x_{n-1} \to x_n). \tag{2.8}$$

The probability $p(x_i \to x_{i+1})$ then depends on the type of the vertices x_i and x_{i+1} (a vertex on a surface with specular/diffuse BSDF, a vertex on a light source, etc.). For details refer to Veach's doctoral thesis [1997].

2.A.2 Merging techniques

The merging techniques differ from the connection techniques in two ways. First, instead of connecting selected vertices of two subpaths, we merge these vertices into one vertex. This means that if the vertices $x_s^{\mathcal{L}}$ and $x_t^{\mathcal{C}}$ lie in the vicinity of each other, we assume they are in fact identical and thus the path is completed (see Figure 2.8).

The second difference from connection techniques is brute force path reuse, where we try to merge one camera subpath with all light subpaths (or vise versa). Therefore one can generate many more full paths given one camera subpath and thus the efficiency of the algorithm is increased. In practice, one then merges a given vertex of one camera subpath with all vertices of all light subpaths that lie in its vicinity (usually within a sphere of a given radius).

As with the connection techniques, different merging techniques vary in the length of the used light subpath and camera subpath. The probability $p_M^{(s,t)}$ of sampling a full path $\overline{x}$ using the (s,t) merging technique is then

$$p_M^{(s,t)}(\overline{x}) = p_C^{(s-1,t)}(\overline{x})p(x_{s-1}^{\mathcal{L}} \to x_s^{\mathcal{L}})\pi r^2. \tag{2.9}$$

Here we use the probability $p_C^{(s-1,t)}$ of sampling $\overline{x}$ using the $(s-1,t)$ connection technique, since it generates most of the vertices in the similar way. r is then a largest distance in which the two vertices $x_s^{\mathcal{L}}$ and $x_t^{\mathcal{C}}$ lie and are still assumed to be identical for the purpose of the merging technique. The derivation of the above equation as well as more details about the merging techniques can be found in the original work on VCM/UPS [Georgiev et al., 2012, Hachisuka et al., 2012].

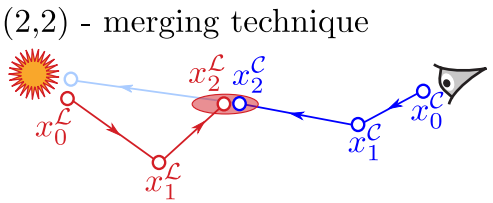

Figure 2.8: We can apply the merging technique to construct a full path from two subpaths if two of their vertices are close enough to each other (these are marked by red ellipse).

3. Introduction to Markov chain Monte Carlo

In this chapter we introduce a class of Monte Carlo sampling algorithms called Markov chain Monte Carlo (MCMC) [Metropolis et al., 1953]. The MCMC algorithms can generate a sequence of samples distributed according to an arbitrary non-negative function (which does not need to be normalized). Therefore they can be effectively used to generate samples from otherwise hard-to-sample distributions. Table 3.1 shows common notation used throughout this section.

$\mathcal{U}$	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 3.1: Common notation.

3.1 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}. 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

$$\pi_i^*(u_i) = \int_{\mathcal{U}} t(v \to u_i)\pi_{i-1}^*(v)\mathrm{d}v. \tag{3.1}$$

[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 transport paths, as discussed below.

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 [Tierney, 1994] 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 [Tierney, 1994].

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 Section 3.6. In the following section, we will focus on a basic MCMC algorithm *Metropolis-Hastings* [Hastings, 1970] and explain the principles in more detail on it. For detailed information about general MCMC, we refer the readers to a more thorough material [Brooks et al., 2011].

3.2 Metropolis-Hastings algorithm

Here we introduce an MCMC algorithm referred to as Metropolis-Hastings [Hastings, 1970]. 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 thesis are derived from it.

The pseudo-code of Metropolis-Hastings (MH) is given in Figure 3.1. 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

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 $u_{i+1} = u_i$

Figure 3.1: Metropolis-Hastings algorithm.

we will accept the proposal is computed (line 4). The probability is defined as

$$\alpha(u \to v) = \begin{cases} \min\left(\frac{\pi(v)q(v \to u)}{\pi(u)q(u \to v)}, 1\right) & \text{if } \pi(u) > 0 \\ 1 & \text{if } \pi(u) = 0 \end{cases} \tag{3.2}$$

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 Section 3.3.

3.3 A simple mutation example

In this section we show 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

$$\pi(u) = \begin{cases} (u - 0.5)^2 & \text{if } u \in [0, 1] = \mathcal{U} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

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 $u_0 = 0$.

First we define a simple uniform mutation Q_{U}

$$Q_{\mathrm{U}}(u, \xi) = \xi, \tag{3.4}$$

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 as discussed in Section 3.4. Histogram (a) in Figure 3.2 shows samples generated using Q_{U}.

We now define another mutation Q_{L}

$$Q_{\mathrm{L}}(u, \xi) = u + \frac{\xi - 0.5}{10}. \tag{3.5}$$

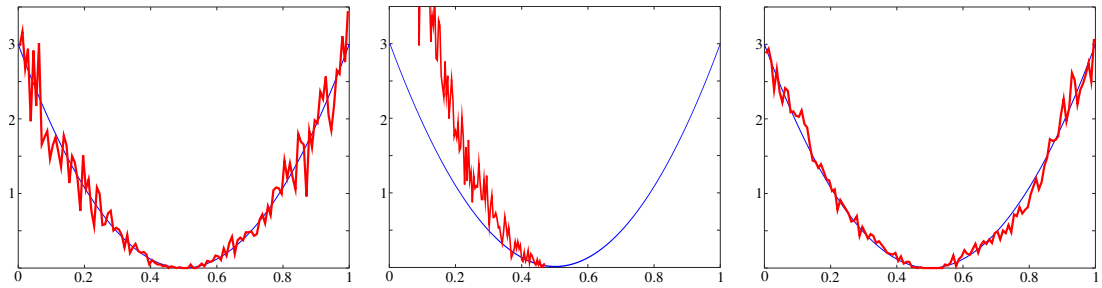

Figure 3.2: This figure shows normalized target function (blue) and histograms of 20,000 samples (red) generated by ten runs of Metropolis-Hastings with different mutations: (a) Uniform independent mutation Q_{U}, (b) local mutation Q_{L} and (c) combination of both mutations Q_{C}. Note that the initial sample u_0 is equal to zero for all runs.

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 (see Section 3.4). Histogram (b) in Figure 3.2 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, in the given number of steps, failed to discover all islands of the state space $\mathcal{U}$ where $\pi(v) > 0$.

Let us now consider a combined mutation Q_{C}

$$Q_{\mathrm{C}}(u, \xi_1, \xi_2) = \begin{cases} Q_{\mathrm{U}}(u, \xi_2) & \text{if } \xi_1 < 0.1 \\ Q_{\mathrm{L}}(u, \xi_2) & \text{otherwise} \end{cases} \tag{3.6}$$

where both ξ_1 and ξ_2 are uniform random numbers. As can be seen from the histogram (c) in Figure 3.2, 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.*

3.4 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

$$t(u \to v) = \begin{cases} q(u \to v)\alpha(u \to v) & \text{if } u \neq v \\ 1 - \int_{\mathcal{U}} q(u \to v)\alpha(u \to v)\mathrm{d}v & \text{if } u = v \end{cases} \tag{3.7}$$

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 Equation (3.7) 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 Section 3.3).

The invariant distribution condition (i.e. the fact that the target distribution is maintained through any transition) can be proven [Veach, 1997] from the fact that Metropolis-Hastings satisfies the so called *detailed balance* for all $u, v \in \mathcal{U}$:

$$\pi(u)q(u \to v)\alpha(u \to v) = \pi(v)q(v \to u)\alpha(v \to u). \tag{3.8}$$

It is important to note that while the *detailed balance* is a sufficient condition, it is not a necessary one (see Appendix 3.A).

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 ($t(u \to u) > 0$) [Nummelin, 1984].

3.5 Using MCMC for quadrature

One of the common uses of MCMC is for numerical integration (quadrature), which is also the case of the previously discussed light transport algorithms (see Section 2). Let us first consider the following integral

$$I = \int_{\mathcal{U}} h(u)\pi(u)\mathrm{d}\mu(u), \tag{3.9}$$

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

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{h(u_i)\pi(u_i)}{p(u_i)} \tag{3.10}$$

where the samples u_i were generated with probability density $p(u_i)$. If $p(u_i) > 0$ whenever $h(u_i)\pi(u_i) > 0$ 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

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{h(u_i)\pi(u_i)}{p(u_i)} = \frac{b}{N}\sum_{i=1}^{N}h(u_i). \tag{3.11}$$

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 it is not advantageous to use MCMC to sample from $h \cdot \pi$. We have to compute b using Monte Carlo with 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 discussed in Section 2.1).

3.6 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* [Szirmay-Kalos et al., 1999]. A common solution applied in computational statistics is to throw away a number of samples [Brooks et al., 2011]. However, it is unclear how many of these samples need to be discarded„ in order to get the start-up bias below some threshold.

Veach and Guibas [1997] proposed a different solution for handling start-up bias when computing quadrature. The solution requires an alternative sampling technique (like bidirectional path tracing in the case of light transport simulation, see Section 2.5) that selects an initial state of the Markov chain. If the initial state u_0 was 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 [Veach and Guibas, 1997] 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

$$b^* = \frac{1}{N} \sum_{i=1} \frac{\pi(u_{0,i})}{p_0(u_{0,i})}. \tag{3.12}$$

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 thesis, from now on we will use the weight b^* instead of the true normalization b in all equations.

3.7 Common optimization

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

Parallelization. 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 [Schmidt et al., 2016]). 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 (see Section 3.3 for an example).

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[Veach, 1997, p. 357]. This technique is an example of Rao-Blackwellization [Blackwell, 1947], 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.

Appendix

3.A Breaking the detailed balance condition

While the detailed balance condition (Equation (3.8)) ensures that a Markov chain is invariant to a given distribution, the condition is not necessary [Manousiouthakis and Deem, 1999]. It is sufficient to satisfy a less strict balance condition, which states that for all $u \in \mathcal{U}$:

$$\int_{\mathcal{U}} \pi(u) q(u \to v) \alpha(u \to v) dv = \int_{\mathcal{U}} \pi(v) q(v \to u) \alpha(v \to u) dv \tag{3.13}$$

The above equation can be translated as how much energy (i.e. target function mass) flows out of any $u \in \mathcal{U}$ must also flow back from all $v \in \mathcal{U}$. It is straightforward to see that the detailed balance (Equation (3.8)) implies the above balance condition.

Breaking detailed balance may potentially result in more efficient mutations [Suwa and Todo, 2010, Manousiouthakis and Deem, 1999], however it is then also more challenging to ensure Equation (3.13) holds. Most methods therefore rather ensure that the detailed balance holds.

4. MCMC algorithms for light transport simulation

In this chapter we give a survey of the state-of-the-art light transport simulation methods that utilize MCMC. While some of the methods were already covered in the recent survey by Jakob and Marschner [2016], here we attempt to give a more complete overview of these algorithms.

We first describe Metropolis light transport by Veach and Guibas [1997], since this algorithm introduced MCMC to light transport simulation. After that we discuss methods that improve the original algorithm. We divide the methods into categories based on which MCMC aspect they try to improve and how they approach that goal.

Prior to discussing the individual MCMC methods, let us briefly summarize the important aspects of an MCMC algorithm that lead to effective 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 (Equation 3.2). This will result in the algorithm "getting stuck" in one mode of the target function (i.e. the chain will generate too many samples in one location before moving to another location), while completely failing to discover other modes. The resulting image may thus miss much of the light transport (see Figure 1.2).

Finally, since light transport simulation estimates multiple integrals, it is important for an MCMC algorithm to ensure **uniform error** reduction. This way the user can clearly see how the image estimation progresses.

4.1 Metropolis light transport

Veach and Guibas [1997] have introduced MCMC to computer graphics in their algorithm Metropolis light transport (MLT). MLT uses Metropolis-Hastings (Section 3.2) 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 $f^*(\overline{x}) = \mathrm{lum}(f(\overline{x}))$. This leads to the following estimator of the light transport equation

$$\langle I_j \rangle = \frac{Pb^*}{N} \sum_{i=1}^{N} \frac{h_j(\overline{x}_i) f(\overline{x}_i)}{f^*(\overline{x}_i)}, \tag{4.1}$$

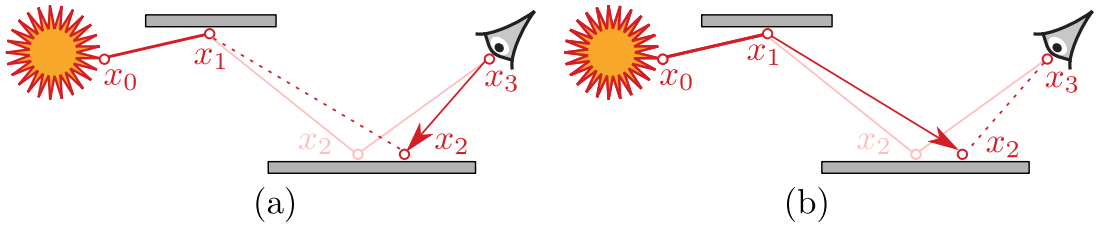

Figure 4.1: In Metropolis light transport (Section 4.1), the lens perturbation (a) creates a new path by changing the direction from a selected vertex x_3 towards the next vertex closer to a light source x_2, the vertex x_1 is then connected to x_2. The caustic perturbation (b) works in the opposite 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.

where P is the number of pixels and the start-up weight b^* (see Section 3.6) is computed from the set of initial samples using separate Monte Carlo integration (such as bidirectional path tracing, see Section 2.5).

Veach and Guibas have devised three different local mutations (which they refer to as *perturbations*) that are effective at generating various types of paths. 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). Figure 4.1 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 paths containing specular-diffuse-specular interactions, e.g. reflected caustics).

The ergodicity is ensured by the *bidirectional* mutation, which can generate any possible path by effectively replacing 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 at a different pixel. MLT was later improved [Pauly et al., 2000] to better handle participating media and two new perturbations 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. Figure 4.2 compares MLT against bidirectional path tracing (see Section 2.5).

4.2 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 may 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.

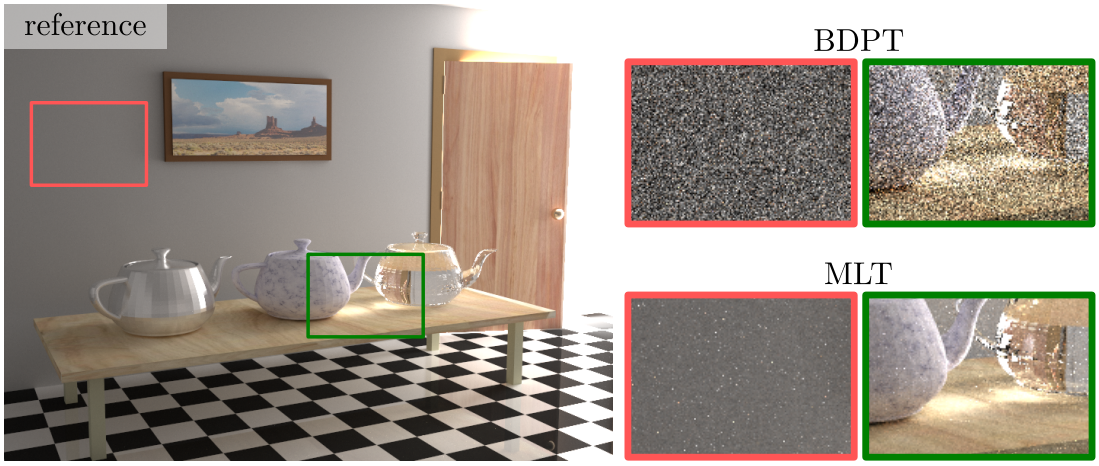

Figure 4.2: Equal-time comparison (12 min.) of the Metropolis light transport (MLT, see Section 4.1) and bidirectional path tracing (BDPT, see Section 2.5). In this case, MLT produces much less noise thanks to its ability to generate more paths that pass through the door ajar and can thus contribute to the image. Scene courtesy of Lehtinen et al. [2013].

4.2.1 Primary sample space Metropolis light transport

Primary sample space MLT (PSSMLT) by Kelemen et al. [2002] utilizes the fact that each light transport 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 (Figure 4.3) given a fixed path sampling technique (Figure 2.4).

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 (Equation (2.1)) that changes the integration domain from the path space Ω to the primary sample space $\mathcal{P}$

$$\begin{aligned} I_j &= \int_\Omega h_j(\overline{x}) f(\overline{x}) \mathrm{d}\mu(\overline{x}) \\ &= \int_\mathcal{P} \frac{h_j(m(\overline{u})) f(m(\overline{u}))}{p_m(m(\overline{u}))} \mathrm{d}\overline{u}. \end{aligned} \tag{4.2}$$

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 $C(\overline{u}) = \frac{f_m(m(\overline{u}))}{p_m(m(\overline{u}))}$, Equation (4.2) can be estimated using MCMC as follows

$$\langle I_j \rangle = \frac{Pb^*}{N} \sum_{i=1}^{N} \frac{h_j(m(\overline{u}_i)) C(\overline{u}_i)}{C^*(\overline{u}_i)}, \tag{4.3}$$

where the target function C^* is the scalar luminosity of C. The start-up weight b^* is computed as discussed in Section 3.6

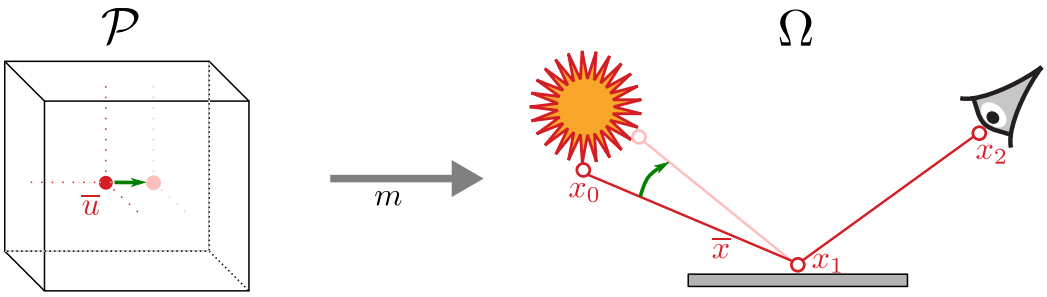

Figure 4.3: A random vector $\overline{u}$ from the primary sample space $\mathcal{P}$ is mapped to a path $\overline{x}$ in the path space. The mapping m is done by sampling vertices of $\overline{x}$ with a given path sampling technique using $\overline{u}$ as random numbers (see e.g. [Pharr et al., 2017, p. 1003] for more details on path sampling). Changing $\overline{u}$ (green arrow) results in a different path $\overline{x}$ being sampled.

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 (Equation (3.2)).[1] The mutation type (small, large) is selected randomly at each step with a probability either given by the user or automatically tuned [Zsolnai-Fehér and Szirmay-Kalos, 2013].

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 (i.e. the path sampling technique) 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 sampling local features, 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.

Utilizing many mappings

We can combine several sampling techniques using MIS (Section 2.3) 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

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

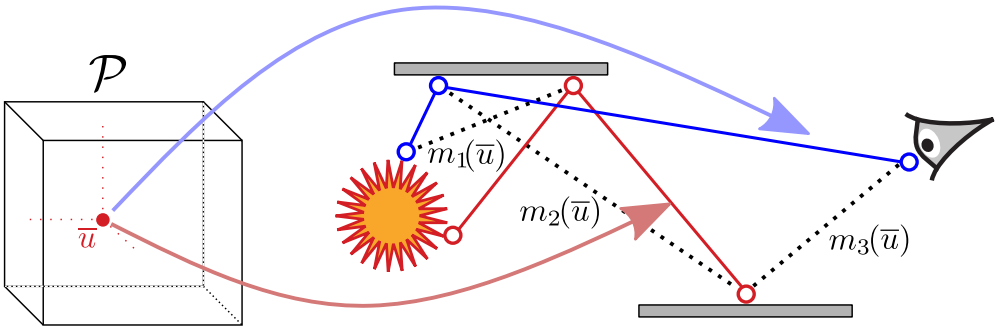

Figure 4.4: Primary sample space MLT (Section 4.2.1) with many mappings treats $\overline{u}$ as a vector of random numbers to generate a camera subpath (blue) and a light subpath (red). The subpaths are combined (dashed line) in various ways yielding several full paths. Each combination corresponds to one mapping m_i.

the contributions of all paths created from a single sample $\overline{u}$

$$C(\overline{u}) = \sum_{m \in \mathcal{M}(\overline{u})} w_m(m(\overline{u})) \frac{f(m(\overline{u}))}{p_m(m(\overline{u}))}, \tag{4.4}$$

where $\mathcal{M}(\overline{u})$ are all the available mappings for a given random vector $\overline{u}$ and w_m is the MIS weight (computed as in bidirectional path tracer [Veach and Guibas, 1994]). With this modification of the definition of $C(\overline{u})$, the final pixel estimator in Equation (4.3) 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 computationally costly.

PSSMLT usually defines its mappings using sampling techniques from bidirectional path tracing (see Section 2.5). 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 (Figure 4.4). 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 [Szirmay-Kalos et al., 2004].

4.2.2 Multiplexed Metropolis light transport

Multiplexed Metropolis light transport (MMLT) [Hachisuka et al., 2014] 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 (Figure 4.5).

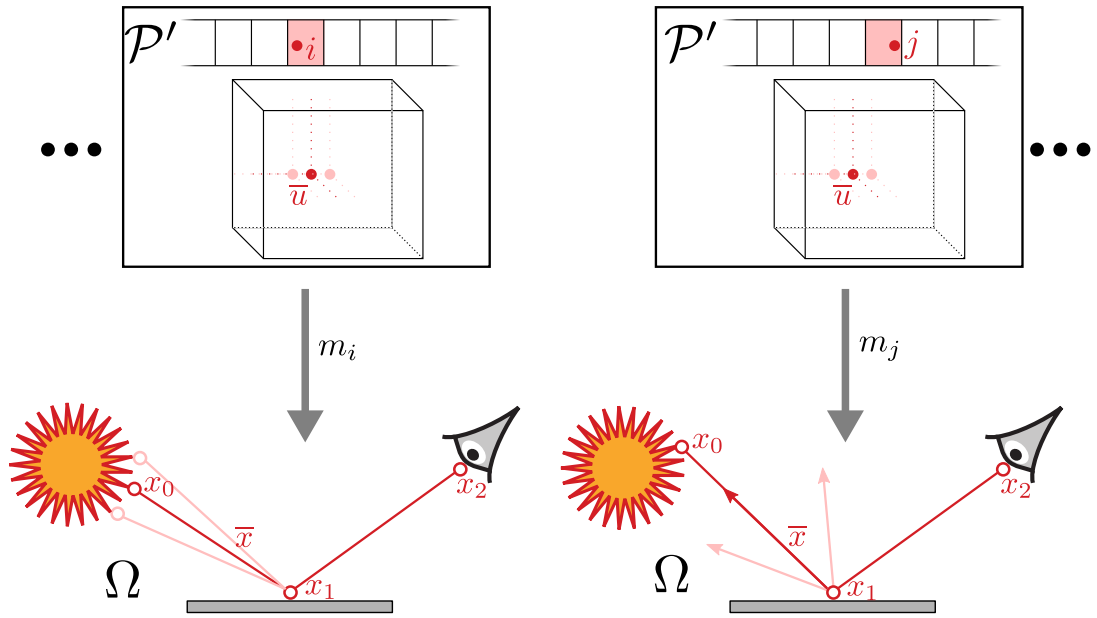

Figure 4.5: Multiplexed MLT (Section 4.2.2) uses the *extended* primary sample space $\mathcal{P}'$, where the first dimension is used to determine a mapping from the rest of $\mathcal{P}'$ to the path space Ω. Here two mappings m_i (next event estimation) and m_j (unidirectional path tracing) map $\overline{u}$ to different paths.

Sampling the mapping

In order to efficiently sample the mapping m_i, MMLT modifies the single mapping target function used in PSSMLT (see Equation (4.3)) by including the MIS weight

$$C_i^*(\overline{u}) = w_{m_i}(m_i(\overline{u}))\frac{f^*(m_i(\overline{u}))}{p_{m_i}(m_i(\overline{u}))}. \tag{4.5}$$

where the mapping m_i is selected from $\overline{u}$ as discussed above. 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 (Equation (4.3)) by using just one mapping per primary sample $\overline{u}$. Furthermore, we need to multiply the estimator by the number of available mappings $\mathcal{M}(\overline{u})$.

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) [Veach, 1997]. 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

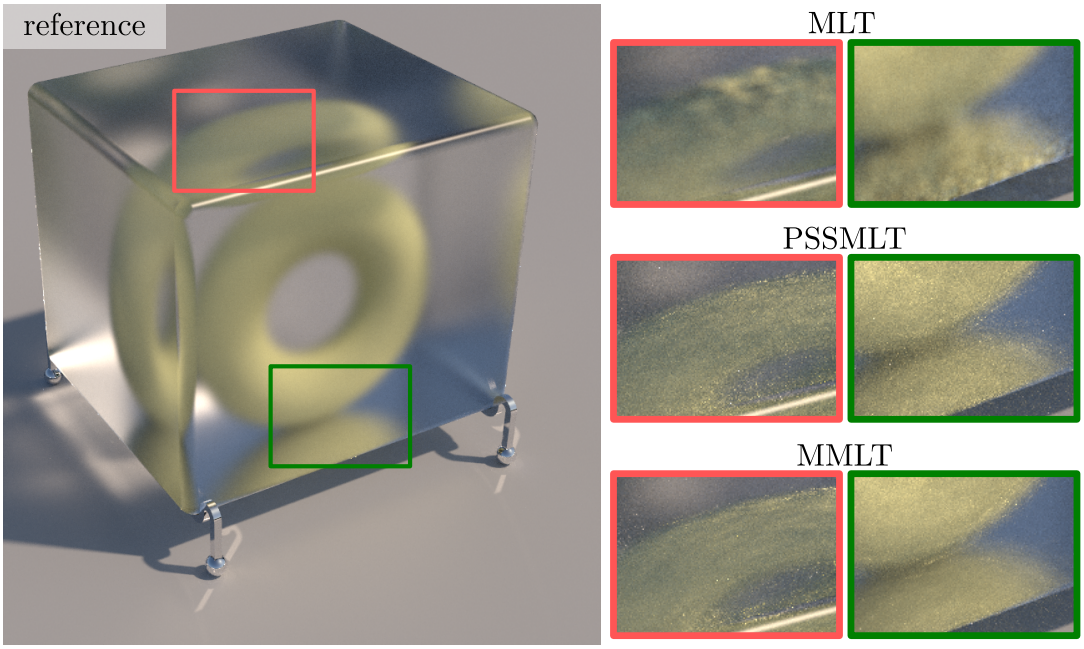

Figure 4.6: The original MLT (Section 4.1) effectively mutates some types of paths, however paths with many specular/glossy interactions are handled poorly. Primary sample space MLT (PSSMLT, Section 4.2.1) maps its samples to the path space using numerous mappings and thus PSSMLT handles effectively more types of paths than MLT. Multiplexed MLT (MMLT, Section 4.2.2) improves upon PSSMLT by using only one mapping per sample. All methods ran for 10 minutes. Scene courtesy of W. Jakob.

a primary sample are possibly shorter and thus PSSMLT can utilize only some of the various mappings and the sample is mapped to fewer paths. In MMLT, one selects a mapping a priori and thus determines subpaths' lengths before generating them. When RR shortens one of the subpaths, MMLT can no longer apply the selected mapping and thus the mutation fails. Figure 4.6 shows a comparison of PSSMLT, MMLT and the original MLT.

4.2.3 Fusing state spaces

Recently, two papers [Otsu et al., 2017, Pantaleoni, 2017] 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 and local exploration of many types of paths, while also utilizing specialized path space mutations (such as those described in Section 4.3) leading to a more robust algorithm (see Figure 4.7 for a comparison). Note that the framework introduced later by Bitterli et al. [2017] allows fusing of state spaces, but the authors have left the actual combination of mutations from both state spaces for future work and apply the framework only for improving MMLT efficiency (see Section 4.2.3).

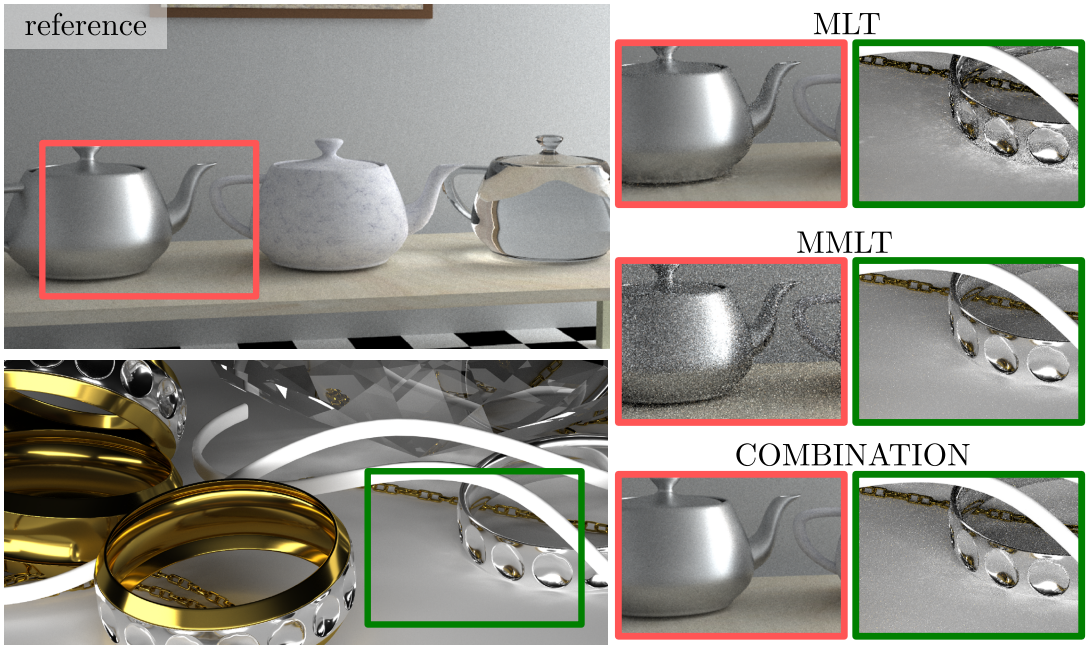

Figure 4.7: Combination of MMLT and MLT mutations (Section 4.2.3) leads to a more robust algorithm, which overall performs on par with the best of the two algorithms in each scene. Image courtesy of H. Otsu et al. [2017].

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 [Otsu et al., 2017, Bitterli et al., 2017].

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. [2017]. 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 [Otsu et al., 2017] for more details.

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 $\overline{u}$ (see Figure 4.5). This may lead to a low acceptance probability. One can utilize the inverse mapping to alleviate this problem [Pantaleoni, 2017, Bitterli et al., 2017]. 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 by the MCMC sampler, 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 the mapping m_j then results in only a small perturbation of $\overline{x}$ in the path space.

4.3 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.

4.3.1 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 [2012] introduced *manifold exploration light transport* (MELT).

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 a 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

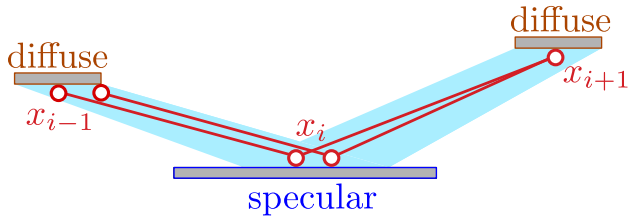

Figure 4.8: The specular manifold (blue), discussed in Section 4.3.1, contains all subpaths created from the original subpath (red) by perturbing position of either x_{i-1} or x_{i+1} (in this example we perturb x_{i-1}), while keeping the constraint c_i^S (Equation (4.6)) on the specular vertex x_i equal to zero, corresponding to fulfilling the law of reflection.

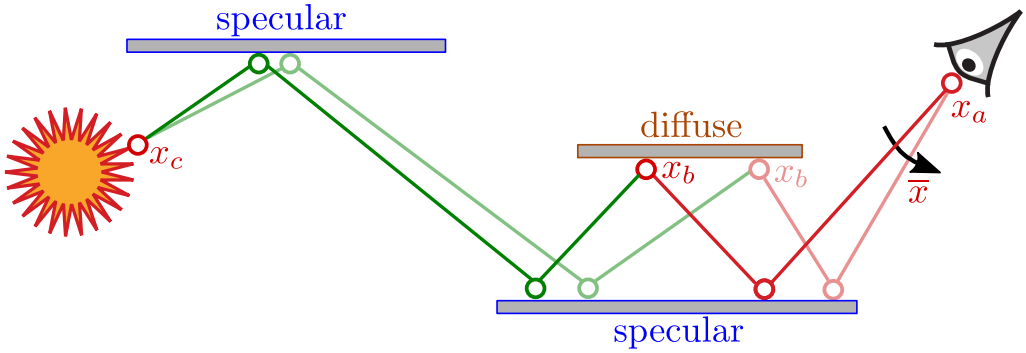

Figure 4.9: The manifold exploration mutation (Section 4.3.1) first changes the direction from x_a and then raytraces the path up to the first diffuse vertex x_b. The manifold walk then updates all specular vertices after x_b up to x_c (green).

in the chain results 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:

$$c_i^S(x_{i-1}, x_i, x_{i+1}) = 0. \tag{4.6}$$

This constraint is equal to zero only if the generalized half-vector [Walter et al., 2007] defined by these vertices is aligned with surface normal, which corresponds to obeying the law of reflection or Snell's law of 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 Figure 4.8.

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 compared to the original MLT.

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 (Figure 4.9). Sample a new direction from x_a and 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. Figure 4.10 shows how the manifold exploration mutation can improve the original MLT algorithm.

To preserve detailed balance Equation(3.8) one must ensure that the mutation is reversible. For this reason, after the path has been updated using the manifold

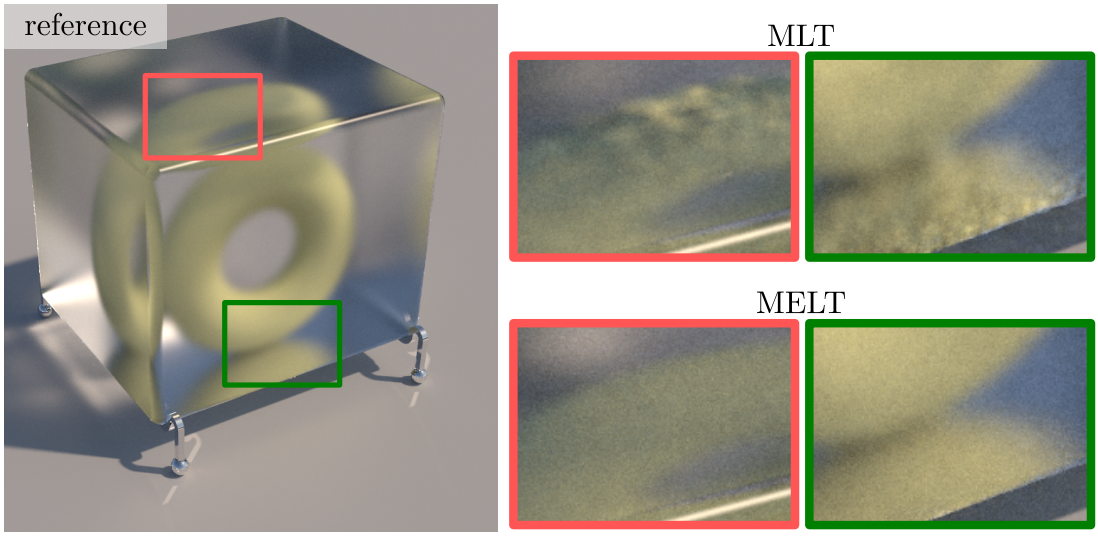

Figure 4.10: The original MLT (Section 4.1) does not handle paths with many glossy/specular vertices effectively. Manifold exploration light transport (MELT, Section 4.3.1) adds a new mutation to MLT that is effective at generating such paths. Both methods were run for 10 minutes. Scene courtesy of W. Jakob.

walk, 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 walk fails or converges to a different than the original configuration (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, there is still room for improvement as shown in the work of Kaplanyan et al. [2014], discussed next.

4.3.2 Half-vector space light transport

Kaplanyan et al. [2014] 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).

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

$$\overline{h} = (x_0, h_1^\perp, \ldots, h_{k-1}^\perp, x_k) \in \Omega(\overline{h}) \subset \Omega. \tag{4.7}$$

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 (Figure 4.11).

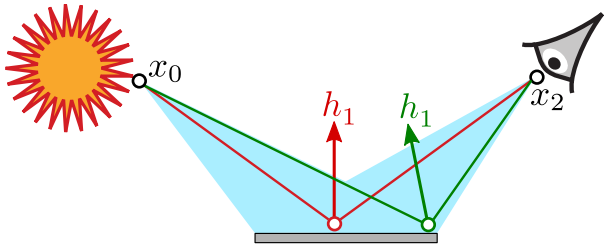

Figure 4.11: The blue area represents the manifold that contains all paths with the end vertices x_0 and x_2 that are uniquely represented by the half-vectors (h_1).

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 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.

Half-vector space mutation

The HSLT mutation takes a path $\overline{h}$ in the NCS and perturbs all the half-vectors resulting in $\overline{h}'$ (Figure 4.12). Once the half-vectors are perturbed, an iterative algorithm similar to the manifold walk (which includes raytracing and computing geometry differentials) 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. Figure 4.13 shows a comparison of HSLT and MELT.

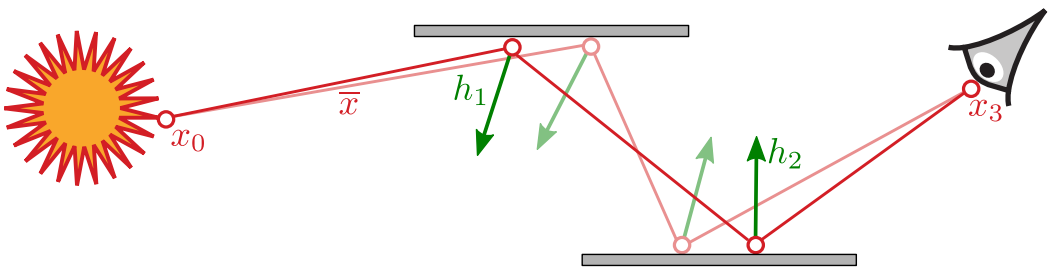

Figure 4.12: Half-vector space mutations (See 4.3.2) perturb all half-vectors (green) of inner vertices at once. This results in a change of position of all inner vertices.

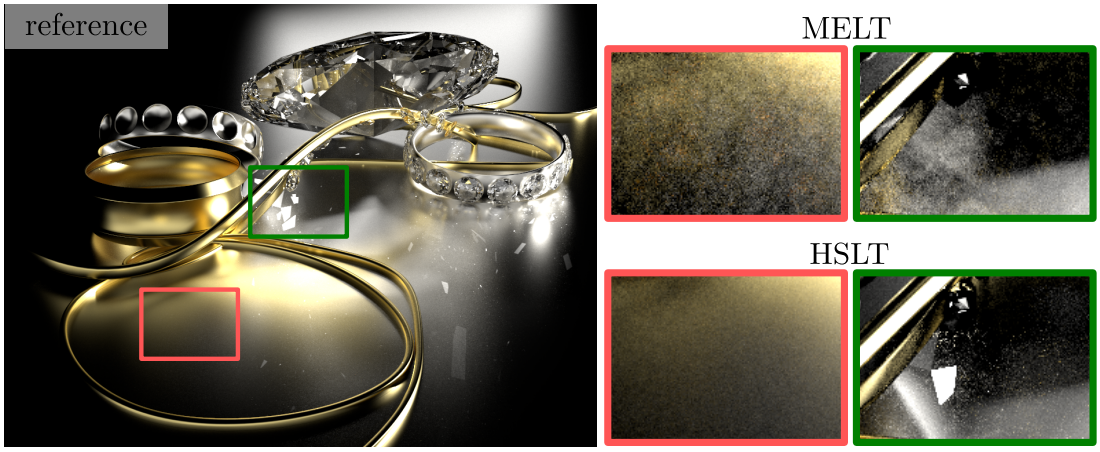

Figure 4.13: Equal time comparison (30 min) of manifold exploration light transport (MELT, Section 4.3.1) and half-vector space light transport (HSLT, Section 4.3.2)) demonstrates that the HSLT mutation is better at local exploration of glossy paths. Scene courtesy of Kaplanyan et al. [2014].

Improved HSLT

The HSLT mutation was later improved [Hanika et al., 2015] 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 $h^{\parallel}$ (Figure 4.14). This allows better sampling of the perturbation, since $h^{\parallel}$ is valid in all $\mathbb{R}^2$ (unlike $h^{\perp}$ which requires clamped or transformed distributions) and corresponds to the domain used for sampling microfacet-based bidirectional scattering distribution 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 iterating/walking over geometrical discontinuities (which often leads to reversibility check failure and the automatic rejection of the mutation).

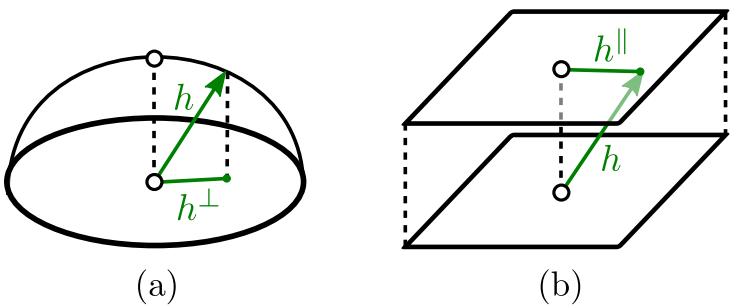

Figure 4.14: Half-vector h can be mapped to a plane using either (a) projection to the tangent plane ($|h^{\perp}| \in [0, 1]$) or (b) plane-to-plane projection ($|h^{\parallel}| \in [0, \infty)$).

4.3.3 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) [Duane et al., 1987]. HMC adapts the mutations according to derivatives of the target function and thus has the potential to achieve better 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 at local maxima. In the following text, we describe the algorithm that introduced HMC to light transport simulation.

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 Appendix 4.A for more details). Li et al. [2015] 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^2MC) algorithm.

To avoid the costly integration, H^2MC locally approximates the target function using Taylor series while utilizing automatic differentiation [Piponi, 2004] (see Appendix 4.A.2). 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 Figure 4.15). The distribution is used to generate new proposals in the primary sample space, which are mapped like in MMLT to the path space (Section 4.2.2).

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. Figure 4.16 shows a comparison of H^2MC with MELT and HSLT. Compared to the previous two methods,

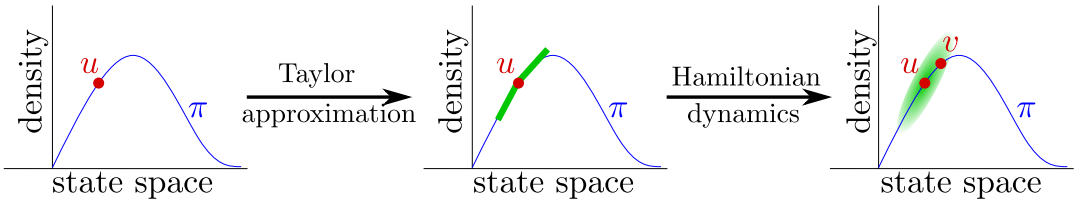

Figure 4.15: In Hessian-Hamiltonian Monte Carlo (H^2MC, Section 4.3.3), the target function π (left) is locally approximated using a Taylor series (green line, middle) around the current sample u. Hamiltonian dynamics are computed using this approximation and result in a Gaussian distribution (green ellipse, right) that is used to generate a proposal v.

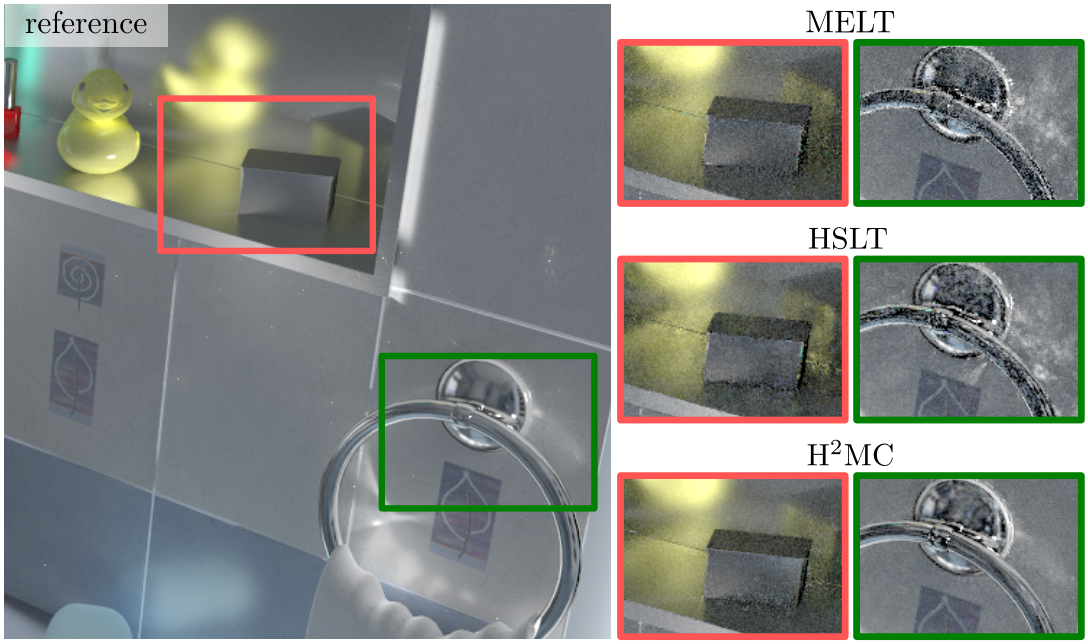

Figure 4.16: Equal-time comparison (10 min) of differential MCMC methods. While manifold exploration light transport (MELT, Section 4.3.1) and half-vector space light transport (HSLT, Section 4.3.2) use geometry differentials to effectively explore some types of paths, Hessian-Hamiltonian Monte Carlo (H^{2}MC, Section 4.3.3) adapts the mutations according to the target function derivatives and thus delivers better performance in this case. Image courtesy of Tzu-Mao Li.

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 unfortunately not always feasible, especially in production renderers).

4.4 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 (Section 4.2.1). 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) [Haario et al., 2001]. 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.

4.4.1 General AMCMC

We give here a brief description of one of the AMCMC methods, *controlled Markov chain Monte Carlo* [Andrieu and Robert, 2001]. For a more comprehensive overview of AMCMC, we refer readers to a survey by Andrieu and Thoms [2008].

Unlike the classical MCMC, in the AMCMC algorithm the transition function t (Section 3.1) 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

$$\theta_{i+1} = H(i, \theta_i, u_0, \dots u_i). \tag{4.8}$$

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 [Andrieu and Thoms, 2008]). To ensure convergence we have to impose a *diminishing adaptation* condition [Andrieu and Thoms, 2008]

$$\lim_{i \to \infty} |H(i, \theta_i, u_0, \dots u_i) - \theta_i| = 0. \tag{4.9}$$

In other words, the adaptation must diminish over time.

4.4.2 Robust adaptive photon tracing

The AMCMC was first introduced to light transport by Hachisuka and Jensen [2011]. Their method improves stochastic progressive photon mapping (SPPM) [Hachisuka and Jensen, 2009], described in the caption of Figure 2.6, by using MCMC to distribute photons into the scene.

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 (Figure 4.17). 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.

Mutation adaptation

The algorithm operates in the primary sample space and adapts the small step mutation (Section 4.2.1). 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

$$\theta_{i+1} = \theta_i + \frac{1}{i}(A_i - A^*) \tag{4.10}$$

If the acceptance rate is below the optimum A^* (23.4% is considered optimal under certain conditions [Rosenthal, 2011]), 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.

Discussion

The resulting algorithm also applies replica exchange (see Section 4.5), however, the second chain effectively only replaces large step mutation since it accepts every uniformly drawn proposal. 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 (Section 4.7.3). The algorithm efficiency could be further improved by adapting the mutations independently for different parts of the state space [Craiu et al., 2009].

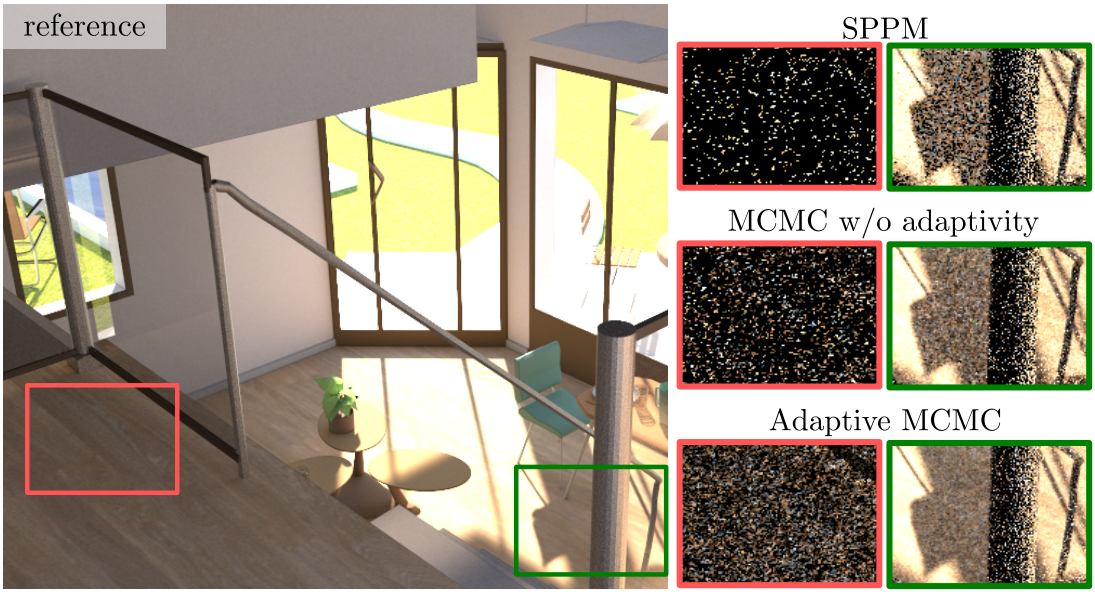

Figure 4.17: The scene is lit from the outside and thus only few photons in stochastic progressive photon mapping (SPPM) [Hachisuka and Jensen, 2009] reach the visible part. While using the method by Hachisuka and Jensen (Section 4.4.2) without adapting the mutations helps to alleviate this problem, the same method with adaptive MCMC is even more efficient. All algorithms ran for 10 minutes.

4.5 Better global exploration I: Tempering and replica exchange

In this section we describe *replica exchange* [Swendsen and Wang, 1986], a general MCMC algorithm that improves robustness of Metropolis-Hastings (Section 3.2). First, we show how replica exchange achieves better *global exploration* than the original Metropolis-Hastings and describe so-called *function tempering*, which further enhances replica exchange. After that we discuss an algorithm that introduced replica exchange to light transport simulation. We further describe work of Kaplanyan and Dachsbacher [2013], which introduces regularization framework. We later show (see Chapter 5) that this regularization can be utilized as function tempering in the path space.

4.5.1 The replica exchange algorithm in general

The replica exchange algorithm (a.k.a. parallel tempering) improves 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 (Figure 4.18). 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 is performed with the probability

$$r(i,j) = \min\left(\frac{\pi_i(u_j)\pi_j(u_i)}{\pi_i(u_i)\pi_j(u_j)}, 1\right), \tag{4.11}$$

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^* [Geyer, 1991].

4.5.2 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 [Earl and Deem, 2005]. One takes the original target function, which is hard to explore and then the function is flattened as follows

$$\pi_i = \pi^{1/T_i}, \tag{4.12}$$

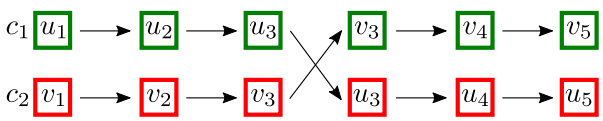

Figure 4.18: Replica exchange (Section 4.5.1) between two chains c_1 and c_2. After several standard mutations their states u_3 and v_3 are exchanged. The chains then again continue mutating these states independently.

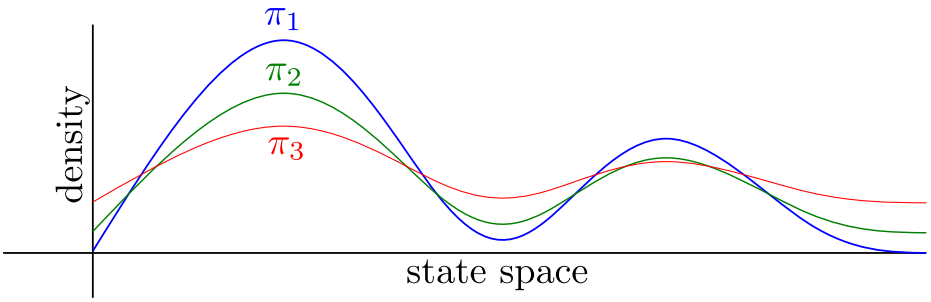

Figure 4.19: The figure shows target functions with increasing temperature. The target function π_1 with the lowest temperature (blue) has the tallest peaks and the lowest valleys and thus is harder to explore than the target function π_3 with the highest temperature (red).

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 (Figure 4.19).

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 $r(i, j)$ is kept high. One also sets T_1 equal to one (i.e. the 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 to easily explore the whole state space and the swap probability is high enough to allow information from high-temperature chains to reach the first chain. One can set the temperatures manually or use automatic methods [Miasojedow et al., 2013].

4.5.3 Replica exchange light transport

Replica exchange was first introduced into light transport in *replica exchange light transport* (RELT) [Kitaoka et al., 2009]. RELT uses several chains that operate in the primary sample space (Section 4.2.1) 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 replica exchange 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[2]. Therefore it behaves similarly as a change of mapping in MMLT (Section 4.2.2) 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 (Section 4.2.3), but it would involve changing the exchange acceptance probability to account for the inverse mapping.

[2]Imagine that one state randomly discovers a path which it cannot 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.

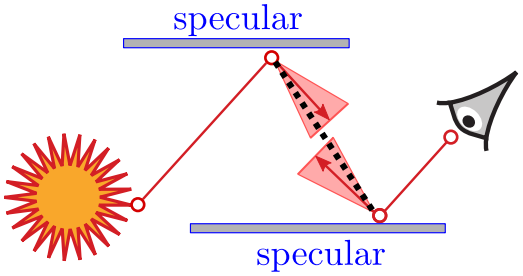

Figure 4.20: Kaplanyan and Dachsbacher (Section 4.5.4) introduced regularization that allows connection of the two specular vertices.

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 at a local mode of the target function is not entirely eliminated.

4.5.4 Path space regularization

To alleviate the problems with the hard-to-sample paths, Kaplanyan and Dachsbacher [2013] introduced the concept of path space regularization. A path that was previously difficult or even impossible to sample due to presence of specular or 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 Figure 4.20. 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 and Dachsbacher further discuss under which conditions will a MCMC-based algorithm converge in the presence of regularization. Figure 4.21 shows a comparison between the original MLT with and without the regularization. In our work (see Chapter 5), we utilize this regularization to apply function tempering in the path space and allow for efficient replica exchange algorithm.

4.6 Better global exploration II: Multiple-try Metropolis

In this section we introduce the *multiple-try Metropolis* (MTM) [Liu et al., 2000] 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 v 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 at 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 (U^* is a set of samples mutated from the final proposal v, see Appendix 4.B

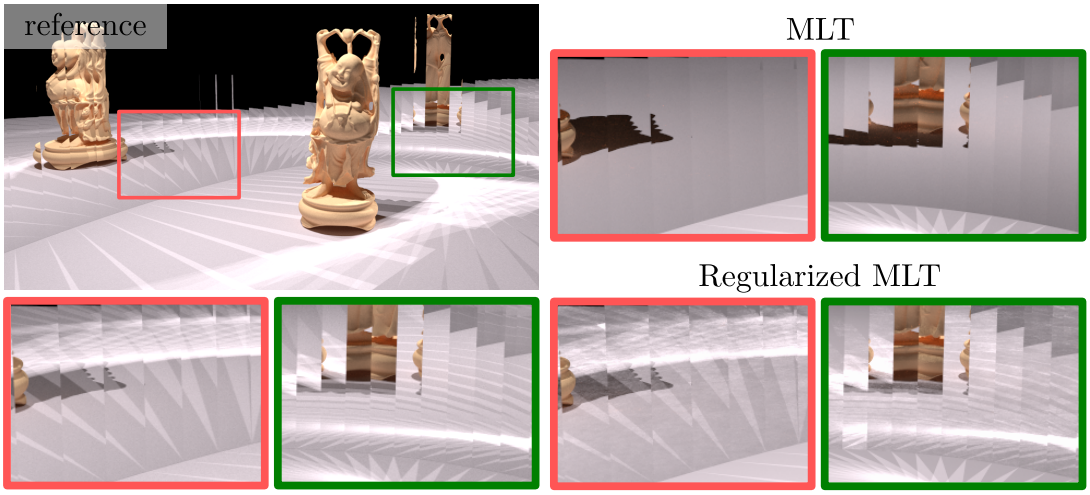

Figure 4.21: Equal-time comparison of the original MLT (MLT, Section 4.1) with and without regularization. The regularization allows the MLT algorithm to discover and sample some paths that were previously under-sampled without the regularization. Image courtesy of Kaplanyan and Dachsbacher [2013].

for more details). In the following text we describe two light transport algorithms that benefit from MTM despite its high computational cost.

4.6.1 Metropolis instant radiosity

MTM was introduced to light transport in an algorithm called *Metropolis instant radiosity* (MIR) [Segovia et al., 2007]. MIR builds on light transport algorithm *instant radiosity* [Keller, 1997], described in the caption of Figure 4.22. 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 (Section 4.1), 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

Figure 4.22: Instant radiosity [Keller, 1997] 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.

Figure 4.23: Multiple-try Metropolis (MTM, Section 4.6) 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.

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. However, the disadvantage is that once the camera moves, the VPLs should be generated again.

Since MCMC samples are commonly heavily correlated, illuminating a scene from a small set of correlated VPLs will lead to serious artifacts. Therefore, the authors have 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). Figure 4.23 shows a comparison of VPL-based methods where VPLs are distributed using MTM and MH.

4.6.2 Coherent MLT

Segovia et al. [2007] 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, Section 4.1) 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 (Section 3.7). Using MTM in this way exploits path coherency, but it also increases sample correlation.

4.7 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, human eyes are similarly sens-

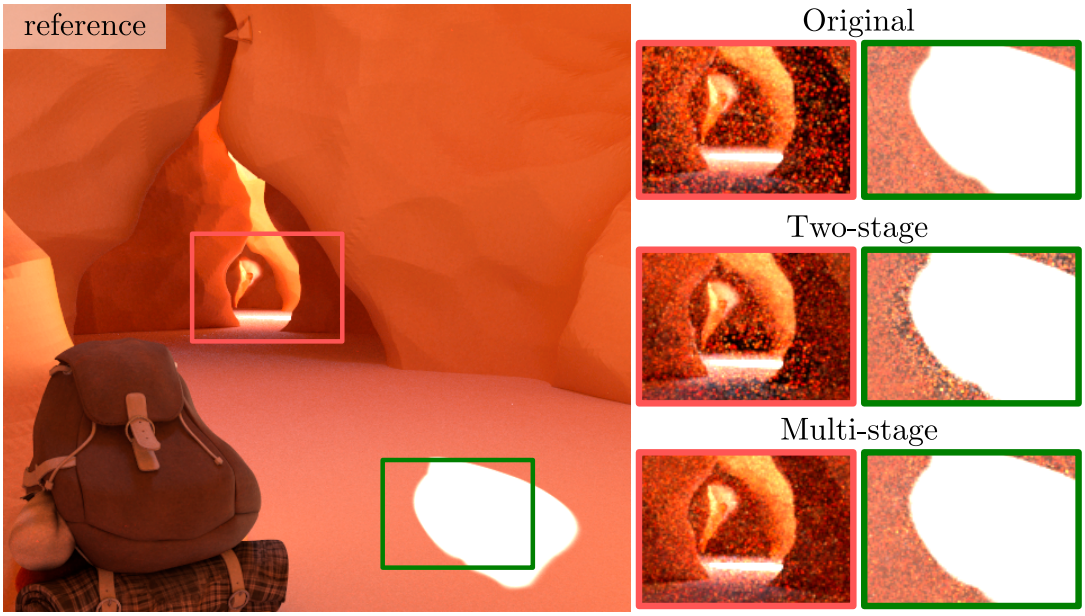

Figure 4.24: Equal-time comparison (30 min) of renderings of a canyon lit through a narrow opening. The original MLT (Section 4.1) under-samples the dark regions. The two-stage method (Section 4.7.1) improves the sample distribution but fails to correctly modify the target function and thus often delivers worse result than the original. The multi-stage version (Section 4.7.1) modifies the target function more robustly and therefore achieves superior results.

itive to relative variance in both bright and dark regions. Thus under-sampling 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.

4.7.1 Two-stage and multi-stage algorithm

Veach in his thesis [1997, 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

$$\pi'(\overline{x}) = \begin{cases} \pi(\overline{x})/I'(\overline{x}) & \text{if } I'(\overline{x}) > 0 \\ \pi(\overline{x}) & \text{otherwise} \end{cases} \tag{4.13}$$

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 under-sampled. Two-stage MLT may therefore converge more slowly than the original MLT [Hoberock and Hart, 2010].

Multi-stage and noise-aware MLT

Hoberock and Hart [2010] improve 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 (Equation 4.13) 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 (Figure 4.24), denoising the first stage without reducing the resolution could potentially work in some cases even better.

Furthermore, Hoberock and Hart 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.

4.7.2 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. [2013] approach improving the distribution of samples in the image differently. They propose *gradient domain MLT* (GDMLT) which builds on manifold exploration light transport (MELT, Section 4.3.1) 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

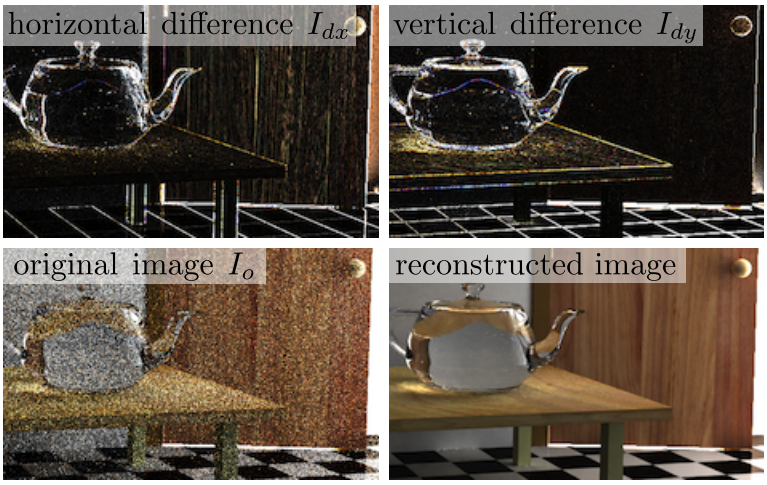

Figure 4.25: Poisson reconstruction computes the final image in gradient domain MLT (Section 4.7.2) from images with finite horizontal I_{dx} and vertical I_{dy} differences and from the original image I_O. Image courtesy of Lehtinen et al. [2013]

.

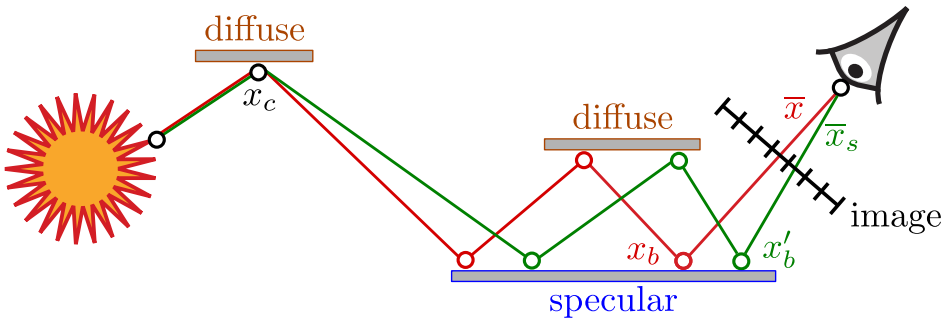

Figure 4.26: Offset path sampling in Gradient domain MLT (Section 4.7.2). The shifted path $\overline{x}_s$ (green) is computed from the original path $\overline{x}$ (red) by first moving the first hit point from the camera x_b to x'_b so that $\overline{x}_s$ goes through a neighboring pixel. The algorithm then generates $\overline{x}_s$ vertices until it can connect to the original path at x_c.

to neighboring pixels. The resulting image is then computed from I_O, I_{dx}, I_{dy} via screened Poisson reconstruction (Figure 4.25). In the case of L^2 Poisson reconstruction [Pérez et al., 2003] GDMLT is an unbiased algorithm as proven by the authors.

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 Figure 4.26. The same paths are used to compute all three images I_O, I_{dx}, I_{dy}. To ensure enough samples is distributed to all important areas of the image (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

$$\pi(\overline{x}, \overline{x}_s) = |f_d^*(\overline{x}, \overline{x}_s)| + \beta \frac{1}{4} f^*(\overline{x}), \tag{4.14}$$

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.

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 Figure 4.25). 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.

Improved sampling for GDMLT

Besides other improvements of GDMLT, Manzi et al. [2014] 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 at a local mode of the target function.

4.7.3 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 (Section 4.4.2).[3] To ensure that image converges uniformly, different approach must be taken for such methods.

Fan et al. [2005] use 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. [2011] enhance the visibility target function used by Hachisuka and Jensen (Section 4.4.2) 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 and Zheng [2015] improve the previous method by utilizing visual importance [Christensen, 2003] 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. [2016] point out that while the other

[3]Since the density estimation radius is gradually shrunk in SPPM (Figure 2.6), the target function of the MCMC algorithms based on SPPM changes during the rendering.

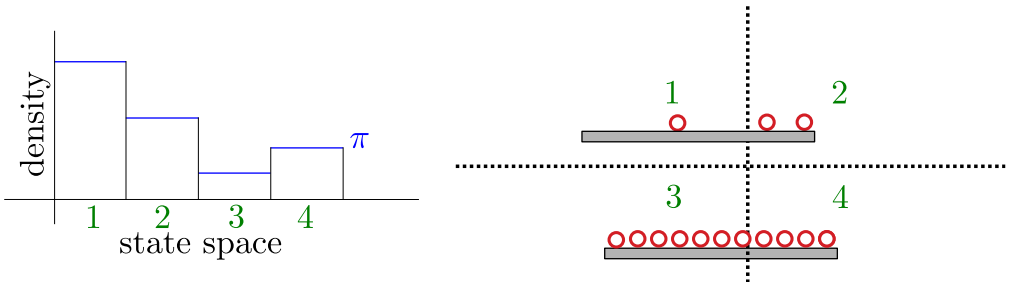

Figure 4.27: According to Gruson et al. (Section 4.7.3), the desired target function for photon tracing is proportional to the inverse of the probability of receiving photon at a given point in the scene. This can be approximated by spatially dividing a scene and counting how many photons (red circles) fall in each bin (marked by green numbers).

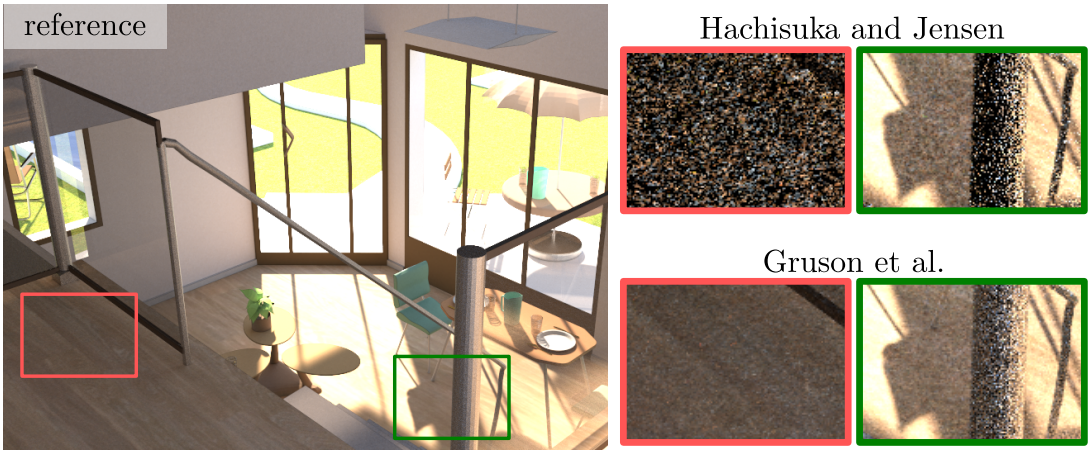

Figure 4.28: Equal-time comparison of algorithms that improve photon guiding. While the method by Hachisuka and Jensen (Section 4.4.2) delivers photons to the visible regions, their distribution is highly non-uniform. The method by Gruson et al. (Section 4.7.3) generates photons in such a way that the resulting image has a more uniform error distribution.

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.

Spatial Target Function

Gruson et al. [2016] 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 (under simplifying assumptions, such as a diffuse BSDF). 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 (Figure 4.27). 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 (Section 4.5). Figure 4.28 shows a comparison with the method by Hachisuka and Jensen (Section 4.4.2).

4.8 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 a method that aims at improving the poor stratification of the usually highly correlated samples generated by MCMC.

4.8.1 Energy redistribution path tracing

Cline et al. [2005] 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 (Section 4.1). 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.

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 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 (Section 4.1), but other path space mutations can be used as well (e.g. manifold exploration, see Section 4.3.1). Each path generated by MCMC then deposits $e_d(\overline{x})$ amount of energy to the image

$$e_d(\overline{x}) = \frac{b^*}{n_c^* n_m}, \tag{4.15}$$

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 (Section 3.6). In order to give unbiased results, each path $\overline{x}$ uses a different number of Markov chains

$$n_c(\overline{x}) = \lfloor \xi + \frac{e(\overline{x})}{n_m e_d(\overline{x})} \rfloor = \lfloor \xi + n_c^* \frac{e(\overline{x})}{b^*} \rfloor, \tag{4.16}$$

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. Therefore, more important initial paths may deposit their energy to more samples and thus decrease the chance of generating an overly-bright pixel.

Similarity to MLT

While ERPT may seem 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 (Equation (4.1)).

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

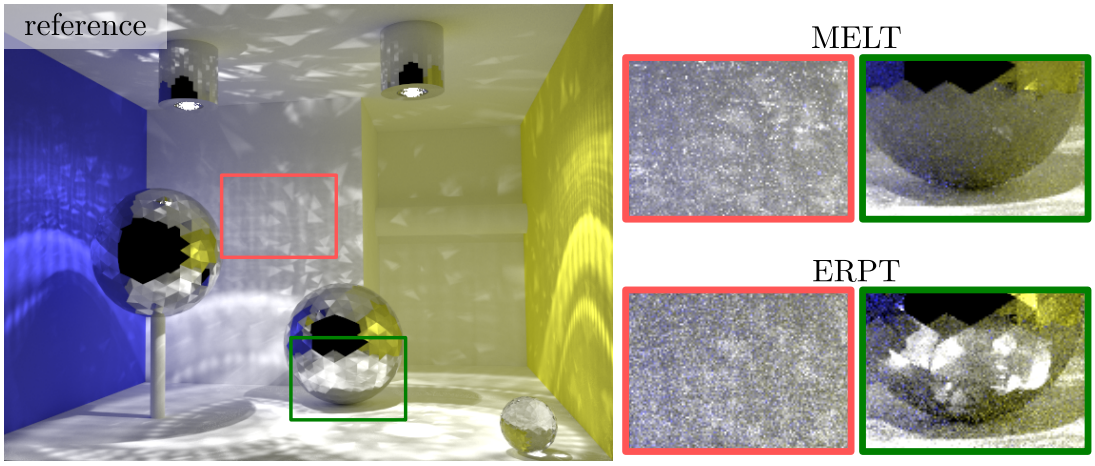

Figure 4.29: Equal-time comparison of manifold exploration light transport (Section 4.3.1) and energy redistribution path tracing (ERPT, Section 4.8.1)with manifold exploration mutation. While MELT delivers less noisy image due to high sample correlation, ERPT discovers more important paths (like reflected caustics) thanks to its better global exploration. Scene courtesy of T. Hachisuka.

unbiasedness of ERPT, but it reduces its efficiency when dealing with hard-to-sample light paths. Figure 4.29 shows a comparison between MLT and ERPT (both utilizing the manifold exploration mutation, see Section 4.3.1).

4.9 Discussion

At the beginning of this chapter, we point out some important aspects of a Markov chain Monte Carlo algorithm: local exploration, global exploration, and uniform image error. Here we compare how different algorithm categories achieve these goals and we offer possible directions for their combination and/or improvement.

4.9.1 Local exploration

Two categories of algorithms aim at improving local exploration. The algorithms discussed in Section 4.2 use a simplified state space called **primary sample space**. A simple mutation in this space (i.e. the small step mutation) 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 Section 4.3, 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 light 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 (Section 4.2.3), it is still unknown how to choose the best mutation from all of the available ones.

In Chapter 6 we discuss our approach to improving both local and global exploration by the means of density estimation along with brute force path reuse. We show that our approach in many cases achieves better results than the complex mutations discussed above.

Since all the discussed mutations depend on user parameters, one can use **adaptive Markov chain Monte Carlo** (Section 4.4) to automatically tune these. While it was shown that this improves the effectiveness of simple mutations in the primary sample space [Hachisuka and Jensen, 2011], 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 [Craiu et al., 2009].

4.9.2 Global Exploration

In this chapter we discuss two general MCMC techniques that improve global exploration: **replica exchange** (see Section 4.5) and **multiple-try Metropolis** (MTM, see Section 4.6). Both of them can be applied in order 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 if properly weighted (as is done in our work, see Section 5.4). The main disadvantage is that its performance highly depends on the number of chains and on their target functions. Tuning replica exchange to run optimally is a difficult task. In Chapter 5 we discuss how we approach the problem of fully utilizing replica exchange in light transport simulation.

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 (see Section 4.6.2), 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 Section 4.6.1); 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** [Mira, 2001], which allows to accept another proposal when the previous one was rejected (without repeating the current sample).

One of the reasons 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 Section 4.2.1). 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 [Jensen, 1995, Vorba et al., 2014] could significantly improve their acceptance rate and thus effectiveness of global exploration.

4.9.3 Uniform image error and stratification

We discuss several methods in Section 4.7 that aim to achieve more uniform image error. Some (Section 4.7.3) are useful only for photon mapping-based methods (Section 2.6), 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 Section 4.8. The energy redistribution path tracing approach (see Section 4.8.1), 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. Using a different approach, as the one discussed in Chapter 6, leads to better results without the need to use short chains.

4.9.4 Error analysis

Variance of the original MLT was analyzed by Ashikhmin et al. [2001] 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 would be to continue the work of Kaplanyan and Dachsbacher [2013], who analyze the mixing rate of a Markov chain (i.e. how fast the chain reaches its stationary distribution)[4].

4.9.5 Temporal stability

Temporal instability is perhaps the thorniest issue of current MCMC-based algorithms and it hinders their adoption in production renderers. The existing Markov chain Monte Carlo methods all exhibit low frequency noise flickering and even appearing and disappearing of whole image features from frame to frame. Ensuring temporal stability comparable to ordinary path tracing is an important topic and we discuss it again in Chapter 6.

[4] In their work, Kaplanyan and Dachsbacher [2013] analyze the mixing rate of a Markov chain to compute bounds for the regularization (see Section 4.5.4).

Appendix

4.A Hamiltonian Monte Carlo

In this appendix we discuss *Hamiltonian Monte Carlo* (HMC) [Duane et al., 1987] in more detail. We first describe the general HMC algorithm and then we describe the details of *Hessian-Hamiltonian Monte Carlo* light transport algorithm [Li et al., 2015] that were left out from Section 4.3.3.

4.A.1 General Hamiltonian Monte Carlo

HMC generates a new proposal by simulating Hamiltonian dynamics over the landscape of the graph of the target function. Hamiltonian dynamics is a system of differential equations defined on Hamiltonian energy $E(u, m)$. Here u represents the position of a point mass and HMC interprets it as the Markov chain state, while m is the mass momentum. The Hamiltonian energy is the sum of potential energy (gravity) $E_U(u)$ and kinetic energy $E_K(m)$

$$E(u, m) = E_U(u) + E_K(m). \tag{4.17}$$

In HMC, the potential energy is defined using the target function π as

$$E_U(u) = -\log(\pi(u)). \tag{4.18}$$

The negative sign ensures that the point mass will be pulled by the gravity towards peaks of the target function π, while log better captures the high dynamic range of π and allows for simpler derivatives of E_U. The kinetic energy is equal to $\frac{1}{2}mAm^T$, where the matrix A transforms the shape of the HMC proposal distribution (A should be optimally set to the covariance of the target function [Neal, 2011]).

To generate a proposal using HMC, we first consider the current state as the position of a point mass and randomly assign to it a momentum m from a Gaussian distribution $\exp(-E_K(m))$ (note that $E_K(m)$ is quadratic in m, hence the previous formula is a Gaussian). Hamiltonian dynamics are then simulated for a fixed time and a new position v and momentum n is obtained. Note that simulating Hamiltonian dynamics for arbitrary target function usually requires time consuming numerical integration. The new state equal to the new position v is then accepted with the probability

$$\alpha(u \to v) = \left(\frac{\pi(v)\exp(-E_K(n))}{\pi(u)\exp(-E_K(m))}, 1 \right). \tag{4.19}$$

Here we have just plugged $\exp(-E_K(m))$ as the mutation kernel into the acceptance probability (Equation (3.2)). Now from Equation (4.17) and Equation (4.18) we directly see that $\pi(u)\exp(-E_K(m)) = \exp(-E(u, m))$ and thus if Hamiltonian dynamics are simulated accurately, the energy remains constant and the proposed v will be always accepted. Due to the potential energy definition in Equation (4.18), Hamiltonian dynamics will create more proposals at places with higher target function value (Figure 4.30) and thus effectively explore local neighborhood of a current sample. For a more comprehensive description of HMC we refer readers to other sources [Neal, 2011].

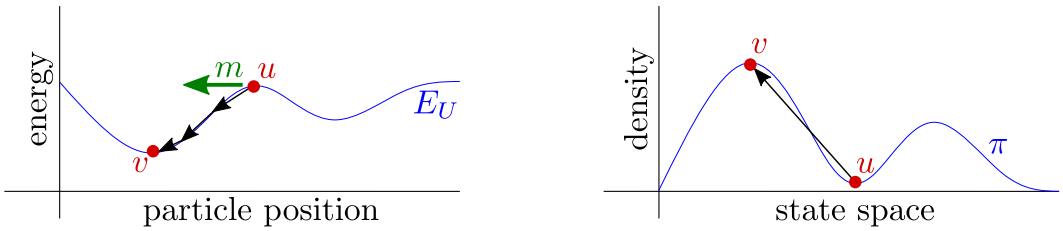

Figure 4.30: In Hamiltonian Monte Carlo (Section 4.A), the potential energy E_U (left) is equal to negative logarithm of the target function π (right). Given a current position (state) u and a randomly sampled momentum m of a point mass, Hamiltonian dynamics are simulated and a new position (proposal) v is obtained. The gravity defined by E_U pulls the point mass downwards, thus the proposal v is pulled towards the peaks of the target function.

4.A.2 Approximating target function in Hessian-Hamiltonian Monte Carlo

As described in Section 4.3.3, *Hessian-Hamiltonian Monte Carlo* (H^2MC) by Li et al. [2015] does not apply HMC directly to avoid the costly integration, which is required to simulate Hamiltonian dynamics. Instead H^2MC locally approximates the target function using a Taylor series

$$\log(\pi(v)) \approx \frac{1}{2} v^T H v + Gv + \log(\pi(u)). \tag{4.20}$$

Here u is the current state and v is a proposal. The Hessian matrix H and the gradient vector G of $\log(\pi(u))$ are computed using automatic differentiation [Piponi, 2004]. This approximation allows Hamiltonian dynamics to be solved analytically.

The authors show that the analytical solution of Hamiltonian dynamics for a new position is actually a linear mapping from momentum m sampled from a Gaussian distribution. Thus a new position (new proposal) is also distributed according to a Gaussian distribution (Figure 4.31).

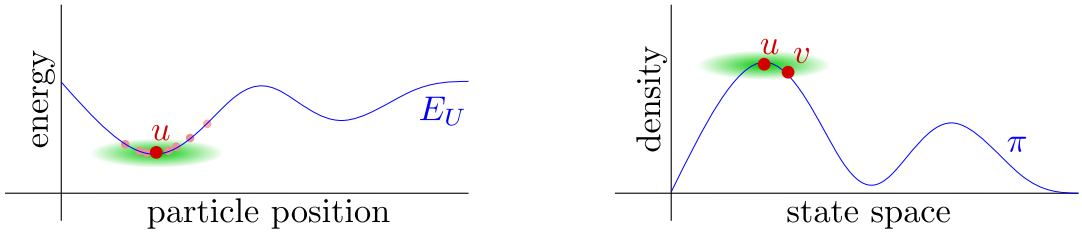

Figure 4.31: In Hessian-Hamiltonian Monte Carlo (H^2MC, Section 4.A.2), the positions generated using Hamiltonian dynamics (left) from the same position u with a momentum sampled from a Gaussian distribution are also distributed according to a Gaussian distribution (green). This distribution is used to generate proposals v in H^2MC (right).

4.B Multiple-try Metropolis

In this section we describe the general *multiple-try Metropolis* (MTM) [Liu et al., 2000] algorithm in more detail. Note that unlike standard Metropolis-Hastings (MH) [Hastings, 1970], the multiple-try Metropolis algorithm tries several proposals before accepting or rejecting a single one of them. The selected proposal should have theoretically higher chance of acceptance compared to MH and thus should decrease the chance of the chain getting stuck at a local maximum.

As in the standard Metropolis-Hastings algorithm, all proposals v are sampled using a mutation Q with the conditional probability $q(u \to v)$. Given a target function π we define the proposal importance weight $\hat{w}$

$$\hat{w}(u \to v) = \pi(v)q(v \to u)\hat{\lambda}(v \leftrightarrow u), \tag{4.21}$$

where $\hat{\lambda}(v \leftrightarrow u)$ is a non-negative symmetric function, which can be chosen by the user. The only requirement is that $\hat{\lambda}(v \leftrightarrow u) > 0$ whenever $q(v \to u) > 0$. $\hat{\lambda}$ influences the probability of accepting a proposed state, however the authors show that its impact is minimal.

The pseudo-code of MTM is outlined in Figure 4.32. The algorithm starts as MH with an initial sample (line 1) from which it generates a Markov chain. Given a sample u_i the algorithm generates k proposals $V = \{v_1, \ldots, v_k\}$ with probability $q(u_i \to v_j)$ (line 3). The final proposal v is then sampled from V with probability proportional to $\hat{w}(u_i \to v)$ (line 4). Next, from the proposal v we sample $k-1$ *competitors* $u_1^*, \ldots, u_{k-1}^*$ with probability $q(v \to u_j^*)$ (line 5). We complete the set of competitors U^* with the current sample $u_k^* = u_i$ (line 6). The acceptance probability with which we will accept the proposal (line 7) is defined as

$$\hat{\alpha}(U^* \to V) = \min\left(1, \frac{\hat{w}(u_i \to v_1) + \ldots + \hat{w}(u_i \to v_k)}{\hat{w}(v \to u_1^*) + \ldots + \hat{w}(v \to u_k^*)}\right) \tag{4.22}$$

Then the proposal is either accepted (line 10) or rejected (line 12) as in standard Metropolis-Hastings.

1: Select u_0
2: **for** $i = 0$ **to** the number of samples **do**
3: Generate k proposals $V = \{v_1, \ldots, v_k\}$
4: Sample a proposal $v \in V$ according to $\hat{w}(u_i \to v)$
5: Mutate v to $k-1$ competitors $u_1^*, \ldots, u_{k-1}^*$
6: Set: $U^* = \{u_1^*, \ldots, u_{k-1}^*, u_k^* = u_i\}$
7: Compute acceptance probability $\hat{\alpha}(U^* \to V)$
8: Generate random number $\xi \sim \mathrm{U}(0, 1)$
9: **if** $\hat{\alpha}(U^* \to V) > \xi$ **then**
10: Accept proposal: set $u_{i+1} = v$
11: **else**
12: Reject proposal: set $u_{i+1} = u_i$

Figure 4.32: Mutliple-try Metropolis algorithm.

From the algorithm description one can immediately observe that for generating a single sample from the Markov chain, $2k - 1$ tentative samples must be generated. Thus the better distribution of samples is payed for by far more expensive sample generation. MTM is therefore mainly useful in applications, where generating fewer but better distributed samples is more advantageous.

5. Improving global exploration by tempering and replica exchange

While many recent papers (see Section 4.3) aim at improving local exploration in MCMC algorithms, the usually poor global exploration in the existing full light transport algorithms remains mostly unaddressed. In this chapter we present a systematic attempt at improving global exploration in MCMC light transport algorithms. We focus on utilizing the replica exchange technique, which achieves better exploration by running several chains with *tempered* target functions and exchanging their states via *replica exchange moves* (see Section 4.5 for more details on replica exchange). While replica exchange has been used previously in light transport simulation (see Section 4.5.3), it has not been fully customized to easily explore all possible light paths and thus has been effective only to some extent. In order to truly unlock the full potential of replica exchange, we must optimize its two key components, tempering and replica exchange moves, considering the specifics of light transport simulation.

We first discuss how to effectively *temper* light transport simulation in order to allow sufficient global exploration of even hard-to-sample paths, such as specular-diffuse-specular ones (see Figure 1.1). We point out that a major source of target function variation are highly glossy or specular bidirectional scattering distribution functions (BSDF) and thus we directly temper them in our approach. This allows for more effective tempering of light transport simulation than using the standard tempering from general MCMC literature [Earl and Deem, 2005, Atchadé et al., 2010].

To maximize the potential of replica exchange, we consider several different *replica exchange moves*. We also design new replica exchange moves that are motivated by the inefficiency of the existing methods. To our knowledge, we are the first to introduce to light transport simulation the equi-energy sampler by Kou et al. [2006] that generalizes replica exchange by allowing exchanges with previously generated states. We compare the different replica exchange moves and the equi-energy sampler in both synthetic and light transport tests to pick the best option for efficient global exploration in light transport simulation.

We further combine the tempering of light transport simulation and the carefully selected replica exchange move in a new MCMC algorithm for light transport simulation. Thanks to its construction, our algorithm can effectively utilize replica exchange to achieve better global exploration compared to the existing MCMC light transport algorithms. The results demonstrate that our algorithm can indeed discover and periodically visit even hard-to-sample paths, see Figure 5.1 for an example.

While our approach certainly improves global exploration in some cases, it is just the first step towards an MCMC algorithm that can easily explore the whole state space and deliver results without any correlation artifacts. We therefore conclude the chapter by discussing other possible approaches that could further improve global exploration.

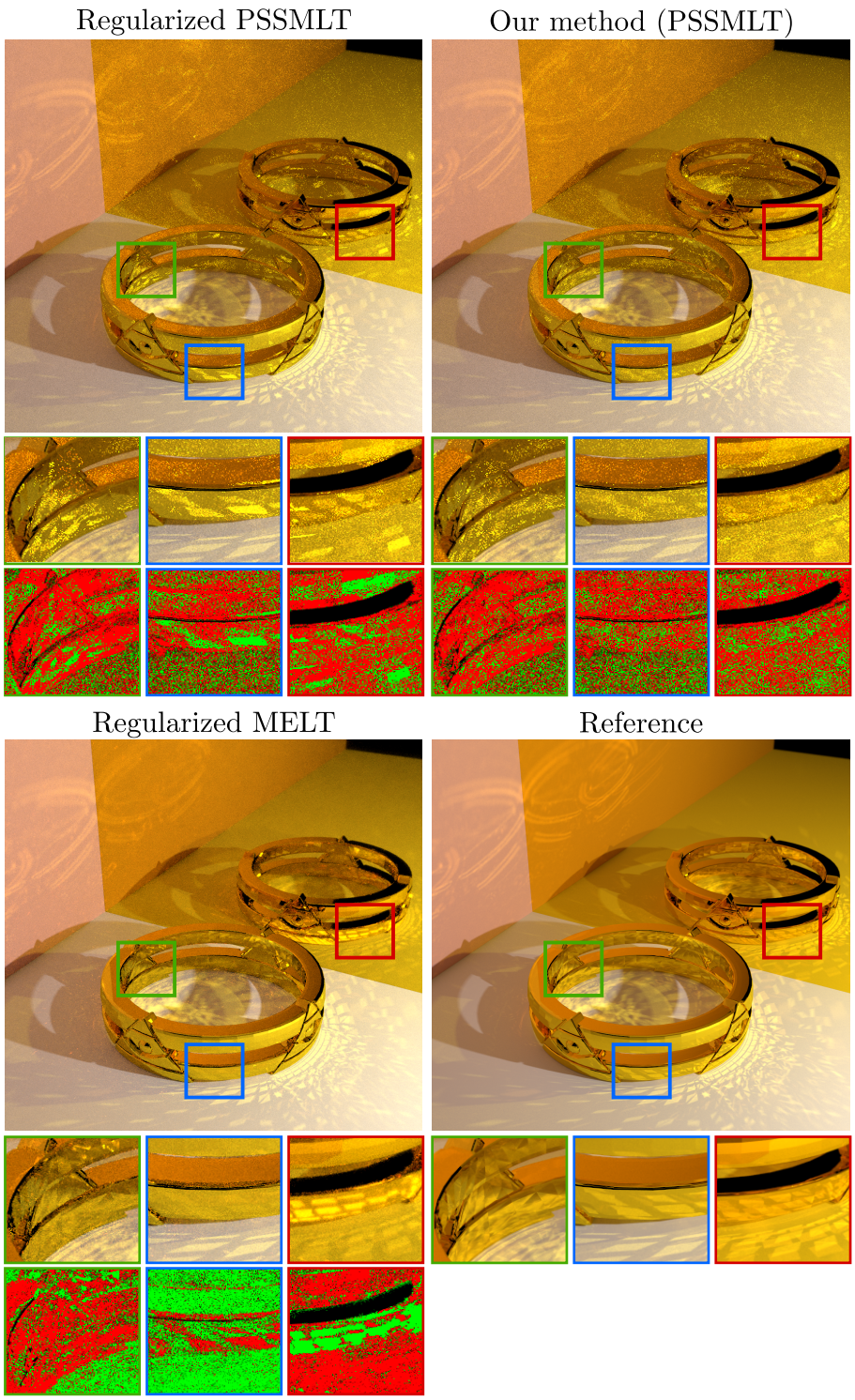

Figure 5.1: Our light transport simulation algorithm improves global exploration through the combination of effective tempering and replica exchange moves. Here we show equal-time comparison (15 minutes) of our method with manifold exploration light transport (MELT, see Section 4.3.1) and primary sample space MLT (PSSMLT, see Section 4.2.1). Note that all the methods utilize path space regularization (see Section 4.5.4) to better handle hard-to-sample paths. While MELT with regularization delivers a cleaner image than our method, it over-samples some parts of the path space, while under-sampling others. The improved global exploration in our algorithm is mostly visible in the positive (green) - negative (red) difference images, where the result of our algorithm exhibits more uniform noise compared to the other tested methods.

5.1 Related work

Replica exchange was introduced to light transport simulation by Kitaoka et al. [2009] in their replica exchange light transport algorithm (RELT, see Section 4.5.3). Thanks to the use of replica exchange, RELT improves the efficiency of primary sample space MLT (see Section 4.2.1), however, RELT cannot sufficiently explore hard-to-sample paths since it does not introduce effective tempering of light transport simulation.

Replica exchange was also applied in some of the MCMC photon tracing algorithms (see Section 4.7.3). In this context, the tempering is directly suited for better placement of photons in the scene, while in our approach we focus on improving global exploration of full light transport paths.

Kaplanyan and Dachsbacher [2013] introduced regularization/tempering of the path space to allow sampling of previously impossible or hard-to-sample paths. The tempering is achieved by widening the lobes of the specular bidirectional scattering distribution functions, see Section 4.5.4 for more details. However, since the tempering cannot be too strong so as to avoid too much bias, this approach does not fully solve the issue of global exploration, as shown in Figure 5.1. In our tempering of light transport simulation we follow the idea of Kaplanyan and Dachsbacher, but we widen the lobes of all glossy and specular BSDFs. Furthermore, more aggressive regularization/tempering is enabled by the replica exchange setting (because the tempered distributions are never directly used for rendering), and thus we achieve better global exploration.

5.2 Tempering of light transport simulation

To enable effective replica exchange, one has to use chains with increasingly flatter target functions. In many physics simulations the complexity of the target function depends on temperature; increasing temperature – *tempering* – of the simulation leads to flatter target functions. We must therefore define how to "increase the temperature" of light transport simulation. Effective tempering of light transport simulation should fulfill the following criteria:

- Increasing the temperature must lead to a flat-enough target function that allows the MCMC algorithm to easily explore the whole state space.
- When the chain with the highest temperature discovers a new part of the state space, the tempering must ensure that this information reaches via state exchanges the main chain (i.e. the chain with the original target function).
- Tempering should be relatively easy to compute, introduce only a low overhead, and should be numerically stable.
- Tempering of light transport simulation should be invariant to rescaling the size of the scene, so the same set of temperatures can be used for all scenes.

In this section, we discuss our approach to tempering of light transport simulation that attempts to satisfy all the above criteria. We monitor the effectivity of the

Figure 5.2: A simple test scene used to measure the effectiveness of function tempering in light transport simulation. We do not use interpolated normals on purpose so that the faceted geometry generates a high number of caustics/reflected caustics. Due to the presence of such hard-to-sample paths, this scene is insufficiently explored by the current MCMC methods: manifold exploration light transport (MELT, see Section 4.3.1) and primary sample space MLT (PSSMLT, see Section 4.2.1). Note that most of the reflected caustics visible in the mirror behind the ring in the reference image are missing in the images generated by MELT and PSSMLT. The reference image was rendered by the VCM/UPS algorithm (see Section 2.7).

tempering on the scene shown in Figure 5.2. While the scene is quite simple, it contains transport paths (e.g. reflected caustics) that cause high peaks in the original target function and thus without tempering the scene is insufficiently explored by the current MCMC methods. In Appendix 5.A we then show other attempts at tempering light transport that we have discovered to be ineffective.

5.2.1 Proposed tempering

In our approach, we concentrate on tempering the target function in the primary sample space (see Section 4.2). The main reason why we choose the primary sample space instead of the path space is because due to inherent importance sampling, it is easier to acknowledge and temper the source of the main variation of the target function. Identifying the exact source is important, since tempering the whole target function is ineffective due to floating point arithmetic inaccuracy (see Appendix 5.A.1).

In the following text we focus on tempering the target function used by primary sample space MLT (PSSMLT, see Section 4.2.1). However, we apply the same tempering on the target function used by multiplexed MLT as well (see Section 4.2.2). We discuss in Section 5.4 that using the tempering in primary sample space MLT leads to an algorithm with better global exploration compared to applying tempering in multiplexed MLT.

The source of the target function variation

To identify the source of the PSSMLT's target function variation we first recall its definition

$$\pi(\overline{u}) = C^*(\overline{u}) = \sum_{m \in \mathcal{M}(\overline{u})} w_m(m(\overline{u})) \frac{\mathrm{lum}(f(m(\overline{u})))}{p_m(m(\overline{u}))}. \tag{5.1}$$

Here m is a path sampling technique (i.e. mapping from the primary sample space to the path space), w_m is the multiple importance sampling weight (MIS weight, see Section 2.3) for a given technique m, $\mathrm{lum}(f)$ is the scalar luminosity of the path contribution function, and p_m is the probability density of sampling a given path $m(\overline{u})$ using a technique m.

Since the MIS weight is limited to the interval $[0, 1]$, we concentrate on the unbounded term $\frac{f(m(\overline{u}))}{p_m(m(\overline{u}))}$, which depends on the path sampling technique m. In PSSMLT each technique $m \in \mathcal{M}$ comes from the bidirectional path tracer (see Section 2.5). The bidirectional path tracer creates a path by importance sampling a subpath from the camera (camera subpath) and a subpath from a light source (light subpath) and connecting their respective end vertices. Therefore the term $\frac{f(m(\overline{u}))}{p_m(m(\overline{u}))}$ can be written as[1]

$$\frac{f(m(\overline{u}))}{p_m(m(\overline{u}))} = C^{\mathcal{L}} f_s(x^{\mathcal{L}}_{s-1} \to x^{\mathcal{L}}_s \to x^{\mathcal{C}}_t) G(x^{\mathcal{L}}_s \leftrightarrow x^{\mathcal{C}}_t) f_s(x^{\mathcal{L}}_s \to x^{\mathcal{C}}_t \to x^{\mathcal{C}}_{t-1}) C^{\mathcal{C}}, \tag{5.2}$$

where $x^{\mathcal{L}}_s$ is the last vertex of the light subpath, $x^{\mathcal{C}}_t$ is the last vertex of the camera subpath, f_s is a bidirectional scattering distribution function (BSDF), G is the geometric factor (see Equation (2.3)), $C^{\mathcal{L}}$ is the throughput of the light subpath (i.e. a product of terms associated with the subpath, like emission, divided by the probability of generating the subpath), and $C^{\mathcal{C}}$ is the throughput of the camera subpath.

Ignoring the inverse squared distance of the geometric factor (that must not be tempered to ensure invariance to scene scaling), the geometric term is limited to the interval $[0, 1]$. While the throughputs of the subpaths $C^{\mathcal{L}}$ and $C^{\mathcal{C}}$ can be a significant source of target function variation, we have find out in our tests that the BSDFs at connecting vertices usually have a greater impact on target function variation, especially in paths with difficult glossy/specular transport. The throughputs $C^{\mathcal{L}}$ and $C^{\mathcal{C}}$ may also contain BSDFs, however, they are usually perfectly importance-sampled and thus they do not have such an impact on the target function. In the following text we therefore concentrate on tempering the BSDFs at connecting vertices. We shortly discuss the remaining sources of variation later in this section.

Tempering the BSDFs

Tempering of BSDFs was already presented by Kaplanyan and Dachsbacher [2013]. In their work, they propose to regularize (temper) the light transport simulation by directly widening the peaks of pure specular BSDFs. This allows

[1] For simplicity, here we do not consider that for some path sampling techniques (e.g. path tracing, light tracing, etc.) some of the terms in Equation (5.2) are missing.

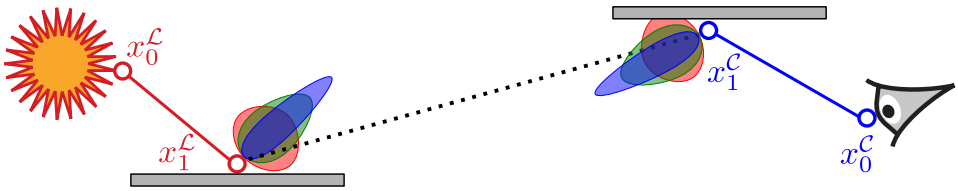

Figure 5.3: We achieve tempering of light transport simulation by widening the BSDF lobe at connecting vertices $x_1^{\mathcal{L}}$ and $x_1^{\mathcal{C}}$. The original blue peak of the BSDF is widened (green, red) and thus the resulting path has higher throughput and is more likely to be discovered and accepted by the MCMC sampler.

easier sampling of previously impossible-to-sample paths (i.e. two subpaths can be connected after the tempering/regularization). Section 4.5.4 discusses their work in more detail.

In our work, we follow the same idea. However, for efficient exploration of the whole path space we continuously temper all possible BSDFs (specular, glossy). It is important to emphasize that we do not temper BSDFs at each path vertex (this would change the simulation too much, see Appendix 5.A.2), instead we temper only the BSDFs at the connecting vertices, see Figure 5.3. We take advantage of the fact that many of the analytic BSDFs used in current rendering systems have a roughness (or glossiness) parameter that controls the width of the BSDF peak. An example of such bidirectional scattering distribution function is the microfacet BSDF with the Beckmann [Ward, 1992] or GGX distribution [Walter et al., 2007], which allows to increase its roughness from completely specular to completely diffuse by changing the distribution of the microfacet normals. We utilize this roughness parameter for BSDF tempering.

More specifically, we derive the tempering of BSDF roughness given the Beckmann distribution $B(m)$ of microfacet normals

$$B(m) = \frac{1}{\pi\rho^2 \cos^4\theta_m} e^{\frac{-\tan^2\theta_m}{\rho^2}}, \tag{5.3}$$

where m is the microfacet normal, θ_m is the angle between the shading normal and the microfacet normal and $\rho > 0$ is the roughness parameter. In general Markov chain Monte Carlo literature [Earl and Deem, 2005], the tempering of the simulation is usually approached by raising the original target function π to the power of the inverse temperature as discussed in Section 4.5.2. We can apply the same principle to the Beckmann distribution

$$B(m)^{(1/T)} = \frac{1}{(\pi\rho^2 \cos^4\theta_m)^{(1/T)}} e^{(1/T)\frac{-\tan^2\theta_m}{\rho^2}} \approx \frac{1}{(\pi(\sqrt{T}\rho)^2 \cos^4\theta_m)} e^{\frac{-\tan^2\theta_m}{(\sqrt{T}\rho)^2}}. \tag{5.4}$$

Thus we arrive to the formula for increasing roughness ρ given a temperature T

$$\rho' = \sqrt{T}\rho. \tag{5.5}$$

Figure 5.4 illustrates that increasing the roughness of BSDFs at connecting vertices indeed allows easier state space exploration.

Note that due to imprecision of floating point arithmetic, directly raising the Beckmann distribution to the power of the inverse temperature instead of roughening would not lead to the same result as shown in Figure 5.24.

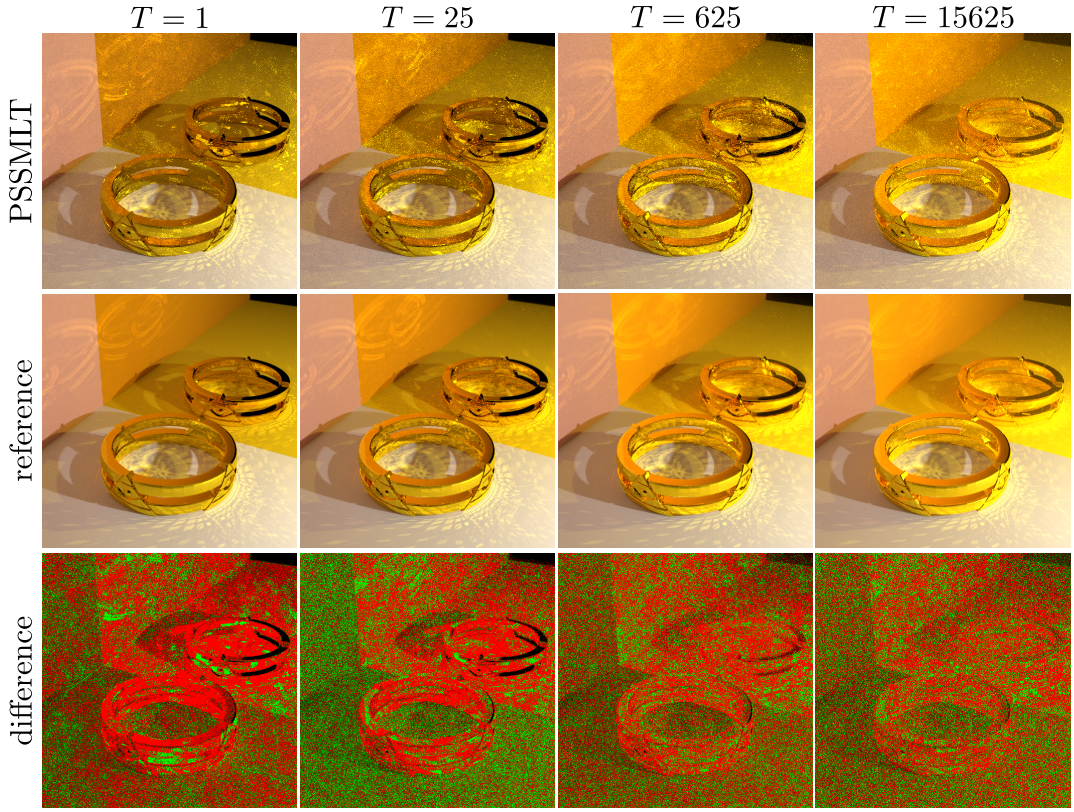

Figure 5.4: Light transport simulation is tempered here by increasing roughness of BSDFs at two vertices connected by deterministic connection as illustrated in Figure 5.3. The original roughness of BSDFs (microfacet BSDF with the Beckmann distribution [Walter et al., 2007]) is tempered according to Equation (5.5). Increasing the roughness allows the primary sample space MLT (PSSMLT, see Section 4.2.2) explore all hard-to-sample paths, which is exhibited by the increasingly more uniform noise in the positive (green) - negative (red) difference images.

From now on we assume that all BSDFs in the scene are either microfacet BSDFs with the Beckmann distribution or diffuse BSDFs which we consider to be easy to explore and thus we do not temper them further. Note that we could also derive tempering for different BSDFs, for example using a similar derivation for the GGX distribution we would get the same result as given in Equation (5.5).

To enable tempering of pure specular BSDFs, we assume they also have a microfacet BSDFs with the Beckmann distribution with the roughness parameter equal to $\rho_{\min} = 10^{-4}$. Note that this assumption is made only when tempering the simulation and not when evaluating the untempered pure specular BSDF (see Section 5.4 for more details about handling of specular BSDFs).

Discussion: The tempered path space

Since we only roughen the BSDFs at connecting vertices, the throughput of a given path $\overline{x} \in \Omega$ depends on the path sampling technique that was used to map a primary space sample to this path. More specifically, it depends on which vertices of the path $\overline{x}$ have a tempered BSDF. If we realize that the original path space can be written as a union of path spaces Ω_k that contain only paths of

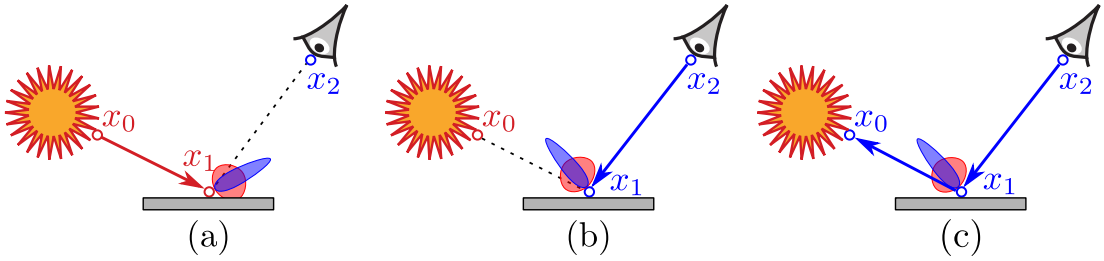

Figure 5.5: A path of length 2 that has one vertex x_1 with tempered BSDF (its lobe is widened from blue to red) can be created by two path sampling techniques: (a) light tracing with next event estimation and (b) path tracing with next event estimation. However, when computing multiple importance sampling (MIS, see Section 2.3) weights, we assume the BSDF at vertex x_1 is always tempered (and not only when x_1 is the end vertex of one subpath) and other path sampling techniques, such as (c) unidirectional path tracing, can create the path as well.

length k [Veach, 1997], $\Omega = \bigcup_{k=1}^{\infty} \Omega_k$, we can define the tempered path space which is sampled in the tempered light transport simulation as follows

$$\Omega' = \bigcup_{k=1}^{\infty} \Omega_k \times (B_k^0 \cup B_k^1 \cup B_k^2). \tag{5.6}$$

Here the set B_k^i holds all the possible positions of vertices with tempered BSDFs in paths of length k with i tempered vertices. More specifically, the set B_k^0 contains exactly one element, so the paths with no tempered BSDFs appear in Ω' exactly once. The two other sets are then defined as $B_k^1 = \{1, \ldots, k-2\}$ and $B_k^2 = \{(1,2), \ldots, (k-3, k-2)\}$. While the tempered path space Ω' has higher dimensionality compared to the regular path space Ω, the tempering allows the MCMC algorithm to efficiently explore even this tempered path space (see Figure 5.4).

We have to consider the tempered path space Ω' also when computing multiple importance sampling (MIS, see Section 2.3) weights of each path sampling technique. For a path $\overline{x} \in (\Omega_k \times B_k^0)$ (i.e. a path with non-tempered BSDFs), we compute the MIS weights as in the standard PSSMLT, considering all path sampling techniques that can create the path $\overline{x}$.

A path $\overline{x}' \in (\Omega_k \times \{j\}) \subset (\Omega_k \times B_k^1)$ that has one vertex x_j' with tempered BSDF can be created by at most two path sampling techniques. Either we connect to the vertex x_j' from the previous vertex x_{j-1} or from the next vertex x_{j+1} in the path. However, considering only those two techniques when computing MIS weights for $\overline{x}'$ significantly increases the target function value for a given primary sample compared to the original target function. Therefore we assume that the BSDF at the vertex x_j' has increased roughness for all PSSMLT path sampling techniques and compute the MIS weight for $\overline{x}'$ accordingly (see Figure 5.5). We compute the MIS weight for a path $\overline{x}'' \in (\Omega_k \times B_k^2)$ that has two vertices with tempered BSDF in a similar fashion as for the path $\overline{x}' \in (\Omega_k \times B_k^1)$.

Note that the MIS weights for the path that has one or two vertices with tempered BSDF often sum up to value lower than one. This influences the result generated by chains that use the tempered path space (i.e. the chains with the tempered target function). As discussed in Section 5.4, we apply additional regularization to handle specular/near-specular BSDFs. In that case, the main chain

with the original target function will also use the tempered path space. Since the regularization is diminished over time, the main chain will after some time sample only the regular path space and thus the algorithm will converge to the correct result [Kaplanyan and Dachsbacher, 2013].

We have also considered using the MIS weights for the tempered path space as they are described by Kaplanyan and Dachsbacher [2013], however in our case (tempering non-specular BSDFs) those MIS weights significantly increased the target function variation. Another option would be to derive MIS weights that consider all possible ways a path $\overline{x} \in \Omega'$ can be created similarly to VCM/UPS [Georgiev et al., 2012, Hachisuka et al., 2012].

Discussion: Remaining sources of variation

So far we have not discussed the other sources of variation of the PSSMLT's target function (Equation (5.1)): MIS weight w_m, the geometry factor G, subpaths' troughputs $C^{\mathcal{L}}$ and $C^{\mathcal{C}}$ (which may contain emission, importance terms, importance-sampled BSDFs), and other terms associated with the BSDFs at connecting vertices that are not influenced by changing the roughness (e.g. the reflectance of BSDF).

In our research we have attempted to further improve global exploration by tempering all these factors by directly raising them to the power of the inverse temperature. While this further increases the ability of the high-temperature chains to discover different modes of the target function faster, it also increases the difference among the tempered target functions. This in turn leads to lower acceptance rate of replica exchange moves and thus worsens global exploration of the main chain. Furthermore, the contribution of samples generated from the other chains to the final image is significantly reduced (we discuss in Section 5.4 how to efficiently combine the contribution from all chains). For this reason we leave these factors untempered.

We must point out that a significant source of the target function variation – visibility between the connecting vertices (which is part of the geometry factor G) – was unaffected by the tested tempering due to the fact that visibility introduces discontinuities that cannot be removed by raising the function to any power. Thus we can only guess if a more sophisticated approach to tempering the visibility would lead to even better global exploration. We leave the tempering of the visibility function for future work.

Discussion: Criteria for effective tempering

Our approach of tempering the light transport simulation by roughening the BSDFs at connecting vertices satisfies all of the criteria for effective tempering stated above. Increasing roughness of BSDFs leads to easier exploration of hard-to-sample paths in scenes that are not dominated by complex visibility as shown in Figure 5.4. Because we directly roughen the BSDFs instead of raising the target function to the power of the inverse temperature, the computation of the tempering is numerically stable. Our tempering is also invariant to scene scaling, since no scale dependent term of the target function is tempered. In the next section, we will show that in our tests the proposed tempering also enables effective

global exploration of the untempered state space (i.e. the information from the highest chain reaches the main chain via state exchanges)

5.3 Replica exchange moves and equi-energy sampler

In this section we attempt to select the best possible replica exchange move for our light transport algorithm. Such a replica exchange move should improve global exploration of the main chain c_1 (i.e. the chain with the original target function $\pi_1 = \pi$) by allowing it to sample the parts of the state space with high target function value that were discovered by the high-temperature chains. It should also propose exchange moves frequently enough and with high acceptance probability (see Equation (4.11)) in order to prevent the main chain c_1 (or other low temperature chains) from getting stuck at a local mode.

We consider here the commonly used replica exchange move, but we also introduce some alternatives as well as our own novel replica exchange moves. Furthermore, we discuss an MCMC algorithm called the equi-energy sampler (see Section 5.3.6) that has not yet been applied in light transport simulation. In order to choose the best option for our algorithm, we compare the different replica exchange moves and the equi-energy sampler in both synthetic scenarios and rendering.

5.3.1 Neighbor swapping

The most common strategy for exchanging the states in replica exchange algorithms is to swap states of the two *neighboring* chains, i.e. the two chains that have the most similar target functions. Exchanging states of the neighboring chains is expected to have high success rate due to the similarity of their target functions. This fact is immediately observable from the exchange acceptance probability in Equation (4.11). Various algorithms that apply this exchange move then only differ in how frequently the swaps are performed [Kitaoka et al., 2009, Gruson et al., 2016]. Here we follow Gruson et al. [2016], who proposed to exchange a chain's state with its neighboring chain's state immediately after chain mutation as illustrated in Figure 5.6. This way the chain has a chance to mutate the state towards a part of the state space with high values of the target function before swapping it to a chain with lower target function value. While neighbor swapping is often a very effective strategy, it has its downsides, as we discuss below.

5.3.2 Importance-sampled permutations

It may often happen that a chain c_j with a high temperature (i.e. a chain that is not a direct neighbor of the main chain c_1) discovers a new mode of the original target function with its current state u_j. In the neighbor swapping algorithm, in order to sample this new mode using the chain c_1 with the original target function, the state u_j must first undergo several successful neighbor swaps to reach the chain c_1. Furthermore, the state u_j must not be mutated outside of

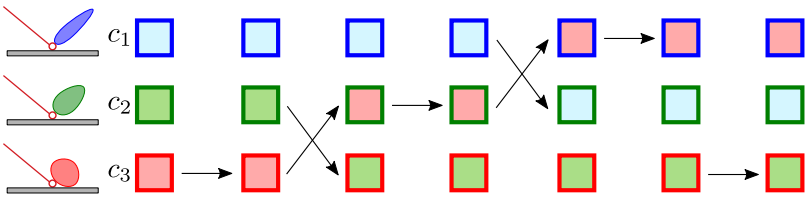

Figure 5.6: Replica exchange with neighbor swapping. Here we consider three chains c_1 (blue border), c_2 (green border), and c_3 (red border) sorted according to increasing temperature of their target function (i.e. chain c_3 has the flattest target function), while their states are represented by the fill color. In this case, neighbor swapping works as follows: we first mutate the chain c_3, attempt a swap between the chains c_3 and c_2, mutate the chain c_2, attempt a swap between the chains c_2 and c_1, and finally mutate the chain c_1. The algorithm then starts again by mutating the chain c_3.

the new mode by any chain before reaching the chain c_1. The probability of such an event is often quite low and thus the mode will remain undiscovered by the chain c_1.

To avoid this issue we would ideally want to immediately swap a state of any chain that has a high value of the original target function to the chain c_1 or other lower temperature chains. To achieve this we propose a new replica exchange algorithm of our own design that utilizes importance-sampled permutations in order to exchange the current states of all chains at once with high probability.

More specifically, given M chains we select a permutation of their states $o : \{1, \ldots, M\} \to \{1, \ldots, M\}$ with probability

$$p(o) = \frac{\prod_{i=1}^{M} \pi_i(u_{o(i)})}{\sum_{o' \in \mathcal{O}} \prod_{i=1}^{M} \pi_i(u_{o'(i)})} = \frac{\prod_{i=1}^{M} \pi_i(u_{o(i)})}{b_{\mathcal{O}}}, \tag{5.7}$$

where $\mathcal{O}$ is a set of all considered permutations, $b_{\mathcal{O}}$ is the probability normalization constant and $u_{o(i)}$ is the state of the chain $c_{o(i)}$ after the permutation. The above probability $p(o)$ is indeed high for a permutation o that moves states with a high value of the target function to the lower temperature chains.

The importance-sampled permutation o is then accepted with the following probability

$$r(o) = \min\left(\frac{\left(\prod_{i=1}^{M} \pi_i(u_{o(i)})\right) p(o^{-1})}{\left(\prod_{i=1}^{M} \pi_i(u_i)\right) p(o)}, 1\right), \tag{5.8}$$

where o^{-1} is the inverse permutation of o (i.e. $o^{-1}(o(i)) = i$) and u_i is the state of the chain c_i prior to the permutation. We have derived that the above acceptance probability leads to a converging MCMC algorithm as discussed in Appendix 5.B.1.

Figure 5.7 demonstrates replica exchange with permutations. Note that since such a permutation can change states of all chains, it can be performed after all chains have been mutated. Now all that remains is to decide which possible permutations $\mathcal{O}$ should we use.

In our tests, we importance-sample a permutation from all possible permutations on the set $\{1, \ldots, M\}$ including the identity. For this complete set of possible

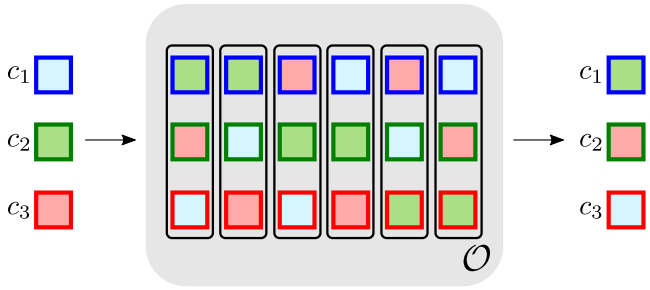

Figure 5.7: A permutation of chain states is importance-sampled with probability given in Equation (5.7) from the set of considered permutations $\mathcal{O}$. Note that the chains are represented by the border color, while their states are represented by the fill colors.

permutations $\mathcal{O}$ the probability of acceptance of an importance-sampled permutation is equal to one as proven in Appendix 5.B.1, and thus the permutation is always accepted. However, selecting the identity (i.e. no changes in any state) has non-zero probability. One could try to use a different set $\mathcal{O}$ with fewer possible permutations (e.g. excluding the identity or including into the set $\mathcal{O}$ only derangements – permutations o that change each state $\forall j \in \{1, \ldots, M\} : o(j) \neq j$). However, such approaches usually lead to a lower acceptance probability and thus we do not include them in our tests.

Note that sampling the permutation in the case of a large number of chains can be very expensive, since there are $M!$ possible permutations of $\{1, \ldots, M\}$. We can, however, take advantage of the fact that the permutation probability in Equation (5.7) is a product of terms, where each term depends on only one permutation input $o(i)$. In Appendix 5.C we present our algorithm that can sample permutations with such a probability in $O(M2^M)$ instead of $O(M!)$, which for $M = 8$ results in practice to roughly 40 times faster algorithm.

5.3.3 Importance-sampled swaps

While thanks to our algorithm we can importance-sample permutations much faster than using a naïve algorithm, we still require exponential time with respect to the number of chains. Furthermore, as the number of chains increases the permutation probability in Equation (5.7) can become numerically unstable for some target functions. To avoid these issues, we propose a novel replica exchange algorithm of our own design that utilizes importance-sample swaps instead of whole permutations.

Given a chain c_i we swap its state with a chain c_j which was importance-sampled from all chains using the following probability

$$p(i, j) = \frac{\pi_i(u_j)\pi_j(u_i)}{\sum_{k=1}^{M} \pi_i(u_k)\pi_k(u_i)}, \tag{5.9}$$

where M is the number of chains. Note that this probability is equal to the permutation probability in Equation (5.7) for $M = 2$. To ensure the algorithm converges, each importance-sampled swap must be accepted with the following

probability (see Appendix 5.B.2 for proof)

$$r_{\mathrm{IS}}(i,j) = \min\left(\frac{\sum_{k=1}^{M} \pi_i(u_k)\pi_k(u_i)}{\sum_{k=1}^{M} \pi_i(u'_k)\pi_k(u'_i)}, 1\right), \tag{5.10}$$

where u'_k is the current state of the k-th chain after the swap

$$u'_i = u_j \wedge u'_j = u_i \wedge \forall k \in \{1,\ldots,M\} \setminus \{i,j\} : u'_k = u_k. \tag{5.11}$$

We must perform the importance-sampled swap before the mutation of any chain, to ensure that the current state of any chain can become the new state of any other chain in just one swap (as in importance-sampled permutations).

Similarly to importance-sampled permutations, importance-sampled swaps can sample identity. This means that when importance-sampling which chain we can swap with the chain c_i, the chain c_i itself can be sampled and thus no swap is actually performed. Excluding such a possibility significantly lowers the acceptance probability and thus brings no advantage.

5.3.4 Equi-energy moves

Baragatti et al. [2012] have shown that the *equi-energy move* (EEM) originally introduced in the equi-energy sampling algorithm (see Section 5.3.6) can also be utilized in a classical replica exchange setting. The idea here is to increase the swap probability (Equation (4.11)) by swapping any two chains that have current states with a similar value of the original target function. To achieve that, the current states of chains are sorted according to their target function values into *energy rings*. Two chains with their current states in the same energy ring can then be swapped with a high probability. Note that unlike in neighbor swapping, here a non-neighboring chains can exchange their states. Thus a part of the state space with high value of the target function discovered by any chain can be sampled by the main chain after just one swap.

More specifically, given a state space $\mathcal{U}$ and an untempered target function π, the authors propose to use a sequence of $d+1$ *energy levels*: $\min_{u\in\mathcal{U}} \pi(u) = H_1 < H_2 \ldots < H_{d+1} = \infty$. The state space $\mathcal{U}$ is partitioned according to energy levels to energy rings $D_i, i = 1,\ldots,d$ as follows:

$$D_i = \{u \in \mathcal{U}; \pi(u) \in [H_i, H_{i+1})\}. \tag{5.12}$$

From the definition of energy levels and energy rings we can see that $\mathcal{U} = \bigcup_{i=1}^{d} D_i$. The partitioning of the state space is illustrated in Figure 5.8.

Replica exchange is then performed by randomly selecting an energy ring D_i that contains current states of at least two chains. From the selected energy ring D_i two chains are uniformly picked and finally their states are swapped.

An important aspect of the algorithm is the definition of energy levels H_i. The authors propose to make them evenly spaced in logarithmic scale (i.e. $H_{i+1}/H_i =$ constant). As for the number of energy rings d, the authors claim that the best results are achieved with d between $M/5$ and $M/3$, where M is the number of chains. However, in our tests we have found out that using approximately the same number of energy rings as the number of chains seems to work the best.

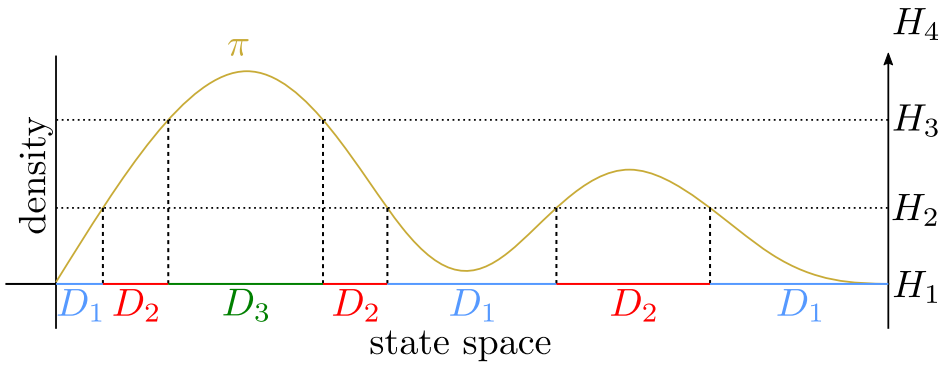

Figure 5.8: The partitioning of the state space to energy rings D_1, D_2, D_3 according to energy levels H_1 to H_4.

Since the proposed equi-energy move can swap any two chains, the authors recommend to apply it always after all chains have mutated their states. This, however, leads to a decrease of the number of attempted swaps compared to neighbor swapping. In the following text, we discuss how to increase the number of attempted swaps in the case of equi-energy moves.

5.3.5 Frequent equi-energy moves

In the original equi-energy moves we swap two selected chains only once all chains have been mutated. This lowers the number of swap attempts compared to other methods and thus may lead to a lower chance of discovering new modes. In order to increase the number of swaps, we propose a novel modification of the algorithm. In our modified version we attempt an equi-energy move before any chain mutation. More specifically, before we mutate the chain c_j, we select the energy ring D_i in which the current state u_j of the chain c_j lies. If the energy ring contains another current state of some other chain, we randomly select chain c_l that has its current state in D_i and is not the chain c_j. We then attempt a swap between the chains c_j and c_l. If the chain c_j has its current state in an energy ring that does not contain any other current state of the remaining chains, we cannot perform the swap. In Appendix 5.B.3 we shortly discuss why this replica exchange move does not break the convergence of the MCMC algorithm.

5.3.6 Equi-energy sampler

In order to ensure a high acceptance probability of the equi-energy move, one must set a sufficiently high number of energy rings. However, in the previous two algorithms using a high number of energy rings also severely decreases the chance of attempting any swap at all (the chance that two chains have their states in the same energy ring is substantially decreased, see Figure 5.9). Kou et al. [2006] have introduced the equi-energy sampler (EES)[2] that circumvents this problem by enabling swaps between the current state of one chain and a past state of another chain. Even with a high number of the energy rings, in EES each

[2]Note that while we describe the energy rings in Section 5.3.4 about equi-energy moves by Baragatti et al. [2012], the energy rings were originally introduced by Kou et al. [2006]

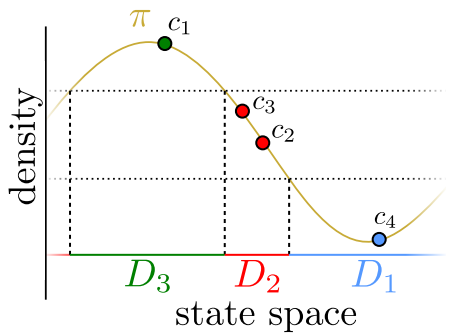

Figure 5.9: Here the chains c_2 and c_3 can be swapped using the equi-energy move, since they both lie in the energy ring D_2. However, the chains c_1 or c_4 have their respective current states in energy rings, where no other chain has its state. Thus neither c_1 nor c_4 can be swapped. Increasing the number of energy rings will result in more chains having their states in different energy rings. Thus the total number of attempted swaps is reduced.

energy ring contains thousands of past samples and thus the algorithm always has exchange candidates with a very high swap acceptance rate.

More specifically, the equi-energy sampler runs M parallel chains, each with a different target function, $\pi_1, \ldots, \pi_M$. For each chain c_j except the chain c_1 (i.e. the chain with the original target function), we prepare an empty energy ring set $\mathcal{D}_{c_j,i}$ for each energy ring D_i. After the burn-in period (i.e. after a given number of samples have been generated), all the chains except the chain c_1 with the original target function start storing all their current samples into energy ring sets. If the current sample u_j of the chain c_j is an element of the energy ring D_i then we store u_j in the corresponding set $\mathcal{D}_{c_j,i}$.

Once each energy ring set of the chain c_j has at least one sample, we can start performing *equi-energy jumps* between chains c_j and c_{j-1}. First, assuming the current state of the chain c_{j-1} lies in the energy ring D_i, a proposal v is randomly selected from the energy ring set $\mathcal{D}_{c_j,i}$. The proposal is then either accepted or rejected using the standard swap acceptance probability (Equation (4.11)). For each chain, the equi-energy jump is used instead of ordinary mutation with a probability p_{EEJ} (authors recommend using $p_{\text{EEJ}} = 0.1$). Note that using the equi-energy jump too often would – due to its high probability of swapping even samples with very high target function value – cause the algorithm to jump to a different state without the chance of proper local exploration.

Energy levels per chain. Using the original target function to define the energy levels may often result in having too many samples generated by high-temperature chains in the first energy ring set (see Figure 5.10). This in turn leads to a lower acceptance rate of swaps between high-temperature chains. To alleviate this problem, we propose to define both energy rings and energy levels for a chain c_j using the target function π_{j-1} of the neighboring chain c_{j-1}. Note

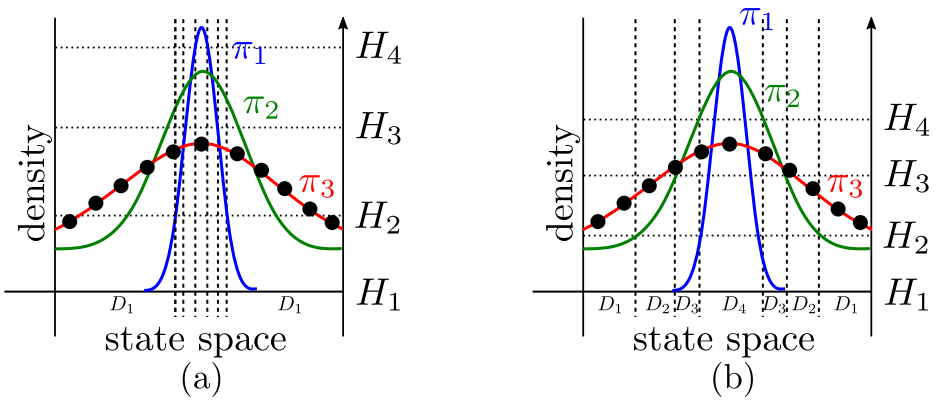

Figure 5.10: (a) Here the majority of samples (black circles) generated by the high-temperature chain c_3 with the target function π_3 (red) are sorted in the first energy ring D_1, since their corresponding original target function π_1 (blue) values are low. Note that the rest of the energy rings are squished in the middle of the state space. Thus an equi-energy jump between the chains c_3 and c_2 (the chain c_2 has the green target function π_2) may have a low acceptance probability.
(b) Here we use the target function π_2 (green) of the chain c_2 to define energy levels and rings for the chain c_3. This results in a better distribution of the samples of the chain c_3 among the energy rings D_i and allows for a higher acceptance probability of equi-energy jumps between the chains c_3 and c_2.

that this trick can be applied in our version of frequent equi-energy moves (see Section 5.3.5) as well[3].

Adaptive equi-energy sampler. To define the energy levels we can apply the adaptive version of the equi-energy sampler [Schreck et al., 2013], where the energy levels for each chain are adaptively changed according to the samples stored in the energy ring sets. The authors recommend to set up energy levels in such a way that each energy ring set accommodates approximately the same number of samples. This can be implemented by sorting all the past samples according to their target function values and then simply split the sorted set into equally sized energy ring sets.

Proof of convergence. It is important to mention that there has been some controversy around the proof of convergence of the equi-energy sampler. Atchadé et al. [2006] have pointed out there were flaws in the original proof by Kou et al. [2006]. Later other works [Andrieu et al., 2008, Hua and Kou, 2011] proved the algorithm convergence under certain conditions. However, since we use the adaptive version of the equi-energy sampler, we can be sure of its convergence due to proof by Schreck et al. [2013].

5.3.7 Synthetic tests

Now that we have covered several different moves for the standard replica exchange and the equi-energy sampler, we test their performance in two simple synthetic scenarios. These scenarios should help us to answer the question which algorithm improves the global exploration better. More specifically, we want to

[3]When attempting equi-energy move with a chain c_j, we use energy rings and levels defined according to its target function π_j.

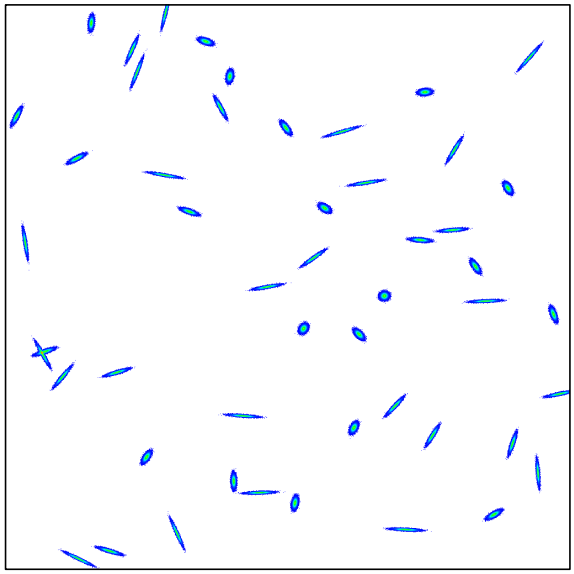

Figure 5.11: A mixture of 50 Gaussians in the unit plane used as a target function in our first synthetic test.

measure how easily the main chain (i.e. the chain with the original target function) discovers new modes of the target function and also how frequently it visits these modes. These scenarios are setup in such a way that they resemble the problem of light transport simulation, while being simple enough to easily measure the performance of different algorithms.

Synthetic scenario setup

Recall that we temper the light transport simulation by roughening the BSDFs (see Section 5.2). More specifically, we assume that a glossy BSDF is defined by the Beckmann distribution of microfacet normals (see Equation (5.3)). The Beckmann distribution is in fact a Gaussian distribution and thus we define here the target function using a mixture of Gaussians.

In the *first scenario*, we use a unit plane as the state space $\mathcal{U} = [0, 1]^2$ and we define the original target function π using a mixture of 50 Gaussians (normal distributions) clamped to fit in $\mathcal{U}$. Each Gaussian is randomly positioned in the unit plane and has a random covariance matrix with limited scale to resemble a highly glossy BSDF lobe, see Figure 5.11. The weights of all the Gaussians in the mixture are set to be equal.

In the *second scenario*, we use a multidimensional state space $\mathcal{U} = [0, 1]^6$ and again define the original target function using a similar mixture of 10 Gaussians. The second scenario should help us see how the algorithms performance deteriorates in higher dimensional state space which is also the case of light transport simulation.

Algorithms and test settings

We hand tune each algorithm to achieve the best performance. Both the original equi-energy moves and our frequent equi-energy moves use 8 energy rings with logarithmically-spaced energy levels. For the adaptive equi-energy sampler we

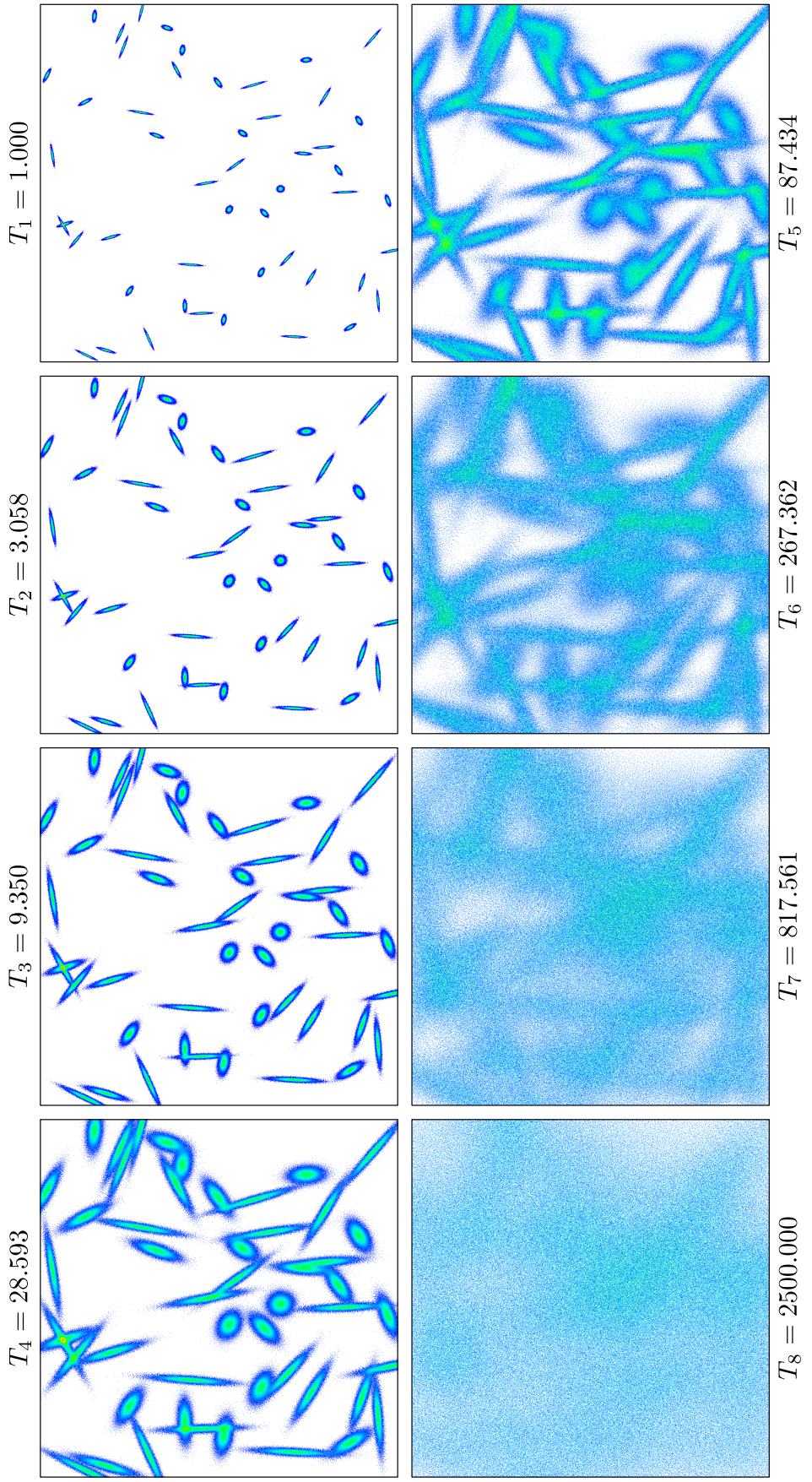

Figure 5.12: Visualization of tempering the target function in the first synthetic scenario discussed in Section 5.3.7.

use 16 energy rings and the probability of equi-energy jump is set to $p_{\text{EEJ}} = 0.1$. Each chain of every tested algorithm is mutated using the small step and large step mutations (with probability of performing the large step mutation equal to 0.3) proposed by Kelemen et al. [2002] for primary sample space MLT (see Section 4.2.1), which corresponds to our final algorithm (see Section 5.4).

Each tested algorithm runs 8 parallel chains and receives the same budget of samples (10000 per chain) excluding the burn-in period of 1000 samples. The chains use target functions with temperatures T_i that span from 1 to 2500 and are evenly spaced on a logarithmic scale (i.e. T_i/T_{i+1} is constant for all i). The highest temperature $T_8 = 2500$ was chosen so that Metropolis-Hastings could easily explore the associated target function π_8. The logarithmic spacing of temperatures then results in roughly similar acceptance rate of neighbor swaps between different neighboring chains [Earl and Deem, 2005]. The tempering is performed by scaling the covariance matrix of each Gaussian by T_i, which corresponds to increasing the BSDF roughness as described in Equation (5.5). Figure 5.12 illustrates the tempering in this scenario.

For each algorithm we measure how well it can estimate the following integral $\int_{\mathcal{U}} \pi(u)^2 \mathrm{d}u$ (the normalization constant needed for the computation of the integral is estimated once and used for all the algorithms). The estimation error gives us an idea whether the algorithm converges at all. More importantly, we measure the difference between the least and the most frequently visited single mode and we also report the number of completely missed modes. This statistic shows us how well the replica exchange algorithm discovers new modes and how frequently it visits them. Note that for all three statistics we consider only the samples generated by the main chain. Each algorithm is then tested in 100 separate runs.

Results

In Tables 5.1 and 5.2 we report the best, the worst, and average results of the 100 runs for the first and the second scenario, respectively. For the first scenario, we further show in Figure 5.13 the distribution of samples generated by each algorithm. Table 5.3 then shows for each algorithm the average chance that a chain swaps its state with another chain.

The first scenario results. From the results, we can clearly see that besides the original Metropolis-Hastings, the worst results are delivered by the adaptive equi-energy sampler (adaptive EES) and the original equi-energy moves (original EEM). We attribute the poor performance of the adaptive EES to the fact that the chains with the higher temperatures are not influenced by the chains with lower temperatures and thus they more often generate samples that are not usable for the lower temperature chains[4]. Poor performance of the original EEM is then due to the fact that the original EEM attempts fewer swaps than the other replica exchange move strategies (seven times fewer swaps than neighbor swapping). All the remaining replica exchange algorithms perform roughly on par. All of them discover all 50 modes and visit each frequently. While neighbor swapping has quite high chance of swapping two chains (53%), its performance is slightly

[4]Allowing equi-energy swaps between all chains (not just neighbors) removes this problem, however the chains then get often stuck at the first few discovered modes, since their samples dominate the energy rings. The results of such modified algorithm are thus even worse.

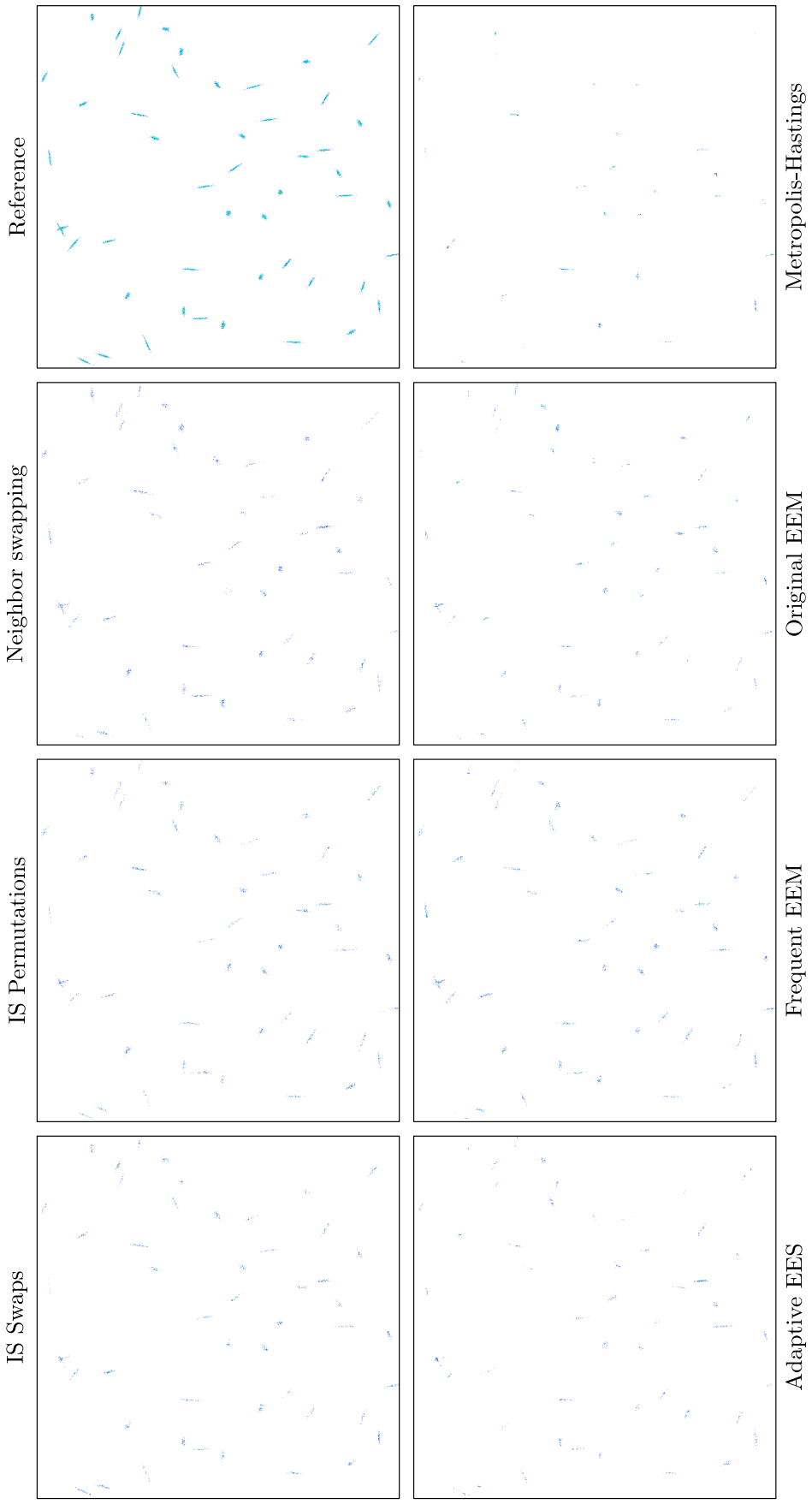

Figure 5.13: Distribution of samples generated by each tested algorithm for the first scenario. The reference was generated by direct sampling of the Gaussian mixture.

BEST RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	0.617%	38.000%	9.910%
Neighbor swapping	0.017%	0.000%	3.260%
IS Permutations	0.027%	0.000%	2.870%
IS Swaps	0.014%	0.000%	3.030%
Original EEM	0.015%	0.000%	3.870%
Frequent EEM	0.037%	0.000%	2.900%
Adaptive EES	0.103%	0.000%	3.880%

AVERAGE RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	5.576%	55.020%	16.702%
Neighbor swapping	2.684%	0.000%	4.732%
IS Permutations	2.358%	0.000%	4.267%
IS Swaps	2.496%	0.000%	4.308%
Original EEM	3.177%	0.260%	5.591%
Frequent EEM	2.558%	0.000%	4.562%
Adaptive EES	3.877%	0.180%	5.939%

WORST RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	18.994%	72.000%	33.930%
Neighbor swapping	9.633%	0.000%	6.990%
IS Permutations	8.799%	0.000%	6.180%
IS Swaps	8.036%	0.000%	7.110%
Original EEM	10.621%	4.000%	10.390%
Frequent EEM	11.188%	0.000%	7.620%
Adaptive EES	13.182%	2.000%	8.610%

Table 5.1: This table shows the best, average and the worst results accumulated over 100 runs of each algorithm for the first scenario (50 Gaussians in two dimensions). For each algorithm we measure the difference in estimation of $\int_{\mathcal{U}} \pi(u)^2 \mathrm{d}u$, percentage of missed modes, percentual difference between the most frequently sampled mode and the least frequently sampled mode (the best result is 0% meaning all the modes are visited an equal number of times). In each category we show the best result in green and the worst result in red (excluding the overall worst results of Metropolis-Hastings).

BEST RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	0.161%	90.000%	100.000%
Neighbor swapping	0.090%	10.000%	21.890%
IS Permutations	0.025%	0.000%	22.250%
IS Swaps	0.116%	10.000%	23.320%
Original EEM	0.407%	10.000%	21.510%
Frequent EEM	0.627%	0.000%	24.910%
Adaptive EES	0.005%	20.000%	35.100%

AVERAGE RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	30.732%	90.000%	100.000%
Neighbor swapping	16.733%	31.400%	46.704%
IS Permutations	14.704%	31.300%	47.045%
IS Swaps	15.094%	35.200%	47.557%
Original EEM	15.060%	41.100%	47.798%
Frequent EEM	16.163%	31.400%	46.352%
Adaptive EES	28.699%	54.600%	68.307%

WORST RESULTS

Algorithm	Estimate diff.	Missed modes	Modes diff.
Metropolis-Hastings	99.389%	90.000%	100.000%
Neighbor swapping	50.224%	70.000%	85.080%
IS Permutations	41.206%	60.000%	83.970%
IS Swaps	54.407%	60.000%	83.420%
Original EEM	57.291%	70.000%	94.060%
Frequent EEM	50.722%	60.000%	95.560%
Adaptive EES	83.415%	80.000%	99.700%

Table 5.2: This table shows the best, average and the worst results accumulated over 100 runs of each algorithm for the second scenario (10 Gaussians in six dimensions). For each algorithm we measure the difference in estimation of $\int_{\mathcal{U}} \pi(u)^2 \mathrm{d}u$, percentage of missed modes, percentual difference between the most frequently sampled mode and the least frequently sampled mode (the best result is 0% meaning all the modes are visited an equal number of times). In each category we show the best result in green and the worst result in red (excluding the overall worst results of Metropolis-Hastings).

Algorithm	Scenario 1	Scenario 2
Neighbor swapping	53%	26%
IS Permutations	67%	36%
IS Swaps	38%	17%
Original EEM	56%	22%
Frequent EEM	39%	18%
Adaptive EES	88%	70%

Table 5.3: The average chance that a chain will swap its state with another chain. Note that when computing this probability we have also considered cases where no swap was possible (original EEM, frequent EEM and adaptive EES) or no swap occurred due to importance sampling of identity (IS permutations and IS swaps).

diminished due to the fact that it cannot swap other chains than the neighboring ones. Importance-sampled (IS) permutations and swaps perform better than neighbor swapping due to their ability to swap any two chains with quite high probability. Note that while a permutation is always accepted, the average chance that a given chain changes its state is 67%. For the IS swaps the average chance that a given chain changes its state is lower (38%), since the swaps are not always accepted, and thus its performance is slightly worse than the performance of IS permutations. Frequent equi-energy moves (frequent EEM) then perform similarly well as importance-sampled swaps, while having 39% chance to perform a swap between any two chains (considering the cases where no move was possible).

The second scenario results. In the second (six-dimensional) scenario, the performance of all algorithms have deteriorated due to the curse of dimensionality. It is much harder to explore a multi-dimensional state space and also the swap probability is significantly lower, see Table 5.3. Here, the worst results are delivered by adaptive equi-energy sampler for the same reasons as in the first scenario. Note that while the original EEM is still the second worst, its performance is now much closer to the other algorithms, since none of the algorithms can really explore this state space well. As for the rest of the algorithms, their performance is almost on par. The best results are delivered by importance-sampled permutations and swaps, and by the frequent equi-energy moves. Again, we hypothesize that this is due to the ability of these algorithms to swap any two chains.

Conclusion. While it is hard to pick a clear winner from the two test scenarios, the importance-sampled permutations perform the best in both scenarios, closely followed by importance-sampled swaps, and frequent equi-energy moves. All three algorithms seem to be most reliable in their ability to discover and frequently visit target function modes. It is also interesting to note that while neighbor swapping is the simplest replica exchange move tested here, it performs almost on par with the other more sophisticated methods. The results of adaptive EES are then surprisingly much worse than those of the standard replica exchange algorithm. While having the highest swap acceptance probability, adaptive EES performance suffers due to its inability to influence the high-temperature chains by the lower temperature chains. In the following text we test the replica-exchange moves and adaptive EES in light transport simulation to decide what should we use for the final version of our light transport algorithm.

5.3.8 Light transport tests

In this section we compare the performance of the different replica exchange moves and the performance of the adaptive equi-energy sampler in light transport simulation. More specifically, we render the ring scene using primary sample space MLT with with the tempering proposed in Section 5.2 and different replica exchange strategies. Section 5.4 discusses the details of this algorithm's implementation.

Unlike in the synthetic tests, here we perform equal-time comparison and thus the Metropolis-Hasting algorithm is able to generate more samples (6% more samples than neighbor swapping). The remaining methods then generate a similar number of samples in a given amount of time (with the adaptive equi-energy sampler having the largest overhead, 7.5% fewer generated samples than neighbor swapping). Note that we utilize all the samples to estimate the image, even those generated by chains with different temperatures (see Section 5.4).

Figure 5.14 shows an equal-time comparison of the different tested variants. We can see from the results that all the tested replica exchange moves lead to better global exploration compared to Metropolis-Hastings. The original equi-energy moves (EEM) deliver the worst result from the tested replica exchange moves due to the low number of attempted swaps. The adaptive equi-energy sampler (EES) then generates the worst result of all the tested methods. We attribute its poor performance to the fact that we were only able to store a limited amount of samples to the energy ring sets due to memory limitation. This in turn caused the algorithm to over-sample the important paths that were stored in the energy rings, while failing to discover remaining important paths. Note that this behavior could have been avoided by reducing equi-energy jump probability, however the resulting algorithm would then behave as Metropolis-Hastings. The results of the remaining methods seem to be roughly on par and thus it is difficult to guess which method performs the best.

To determine the best method, we show a convergence plot in Figure 5.15. Note that we exclude from this plot the adaptive equi-energy sampler (which has the lowest value of RMSE equal to 2.440) to better show the differences among the other methods. From this plot we can observe that the original EEM and Metropolis-Hastings have the worst convergence rate. The remaining methods perform almost on par, with the importance-sampled (IS) permutations and the IS swaps delivering the best results. Since the IS permutations performed the best also in the synthetic tests, we choose this move for our final algorithm.

5.4 Implementation details

Our novel MCMC algorithm combines the discussed tempering of light transport simulation and replica exchange performed by the importance-sampled permutations. We have created two variants, one built on top of primary sample space MLT (see Section 4.2.1) and the other built on top of multiplexed MLT (MMLT, see Section 4.2.2). Note that both versions of our algorithm thus operate in the primary sample space.

Figure 5.16 compares these two variants of our algorithm. While our method built on top of MMLT delivers cleaner result due to better local exploration, it

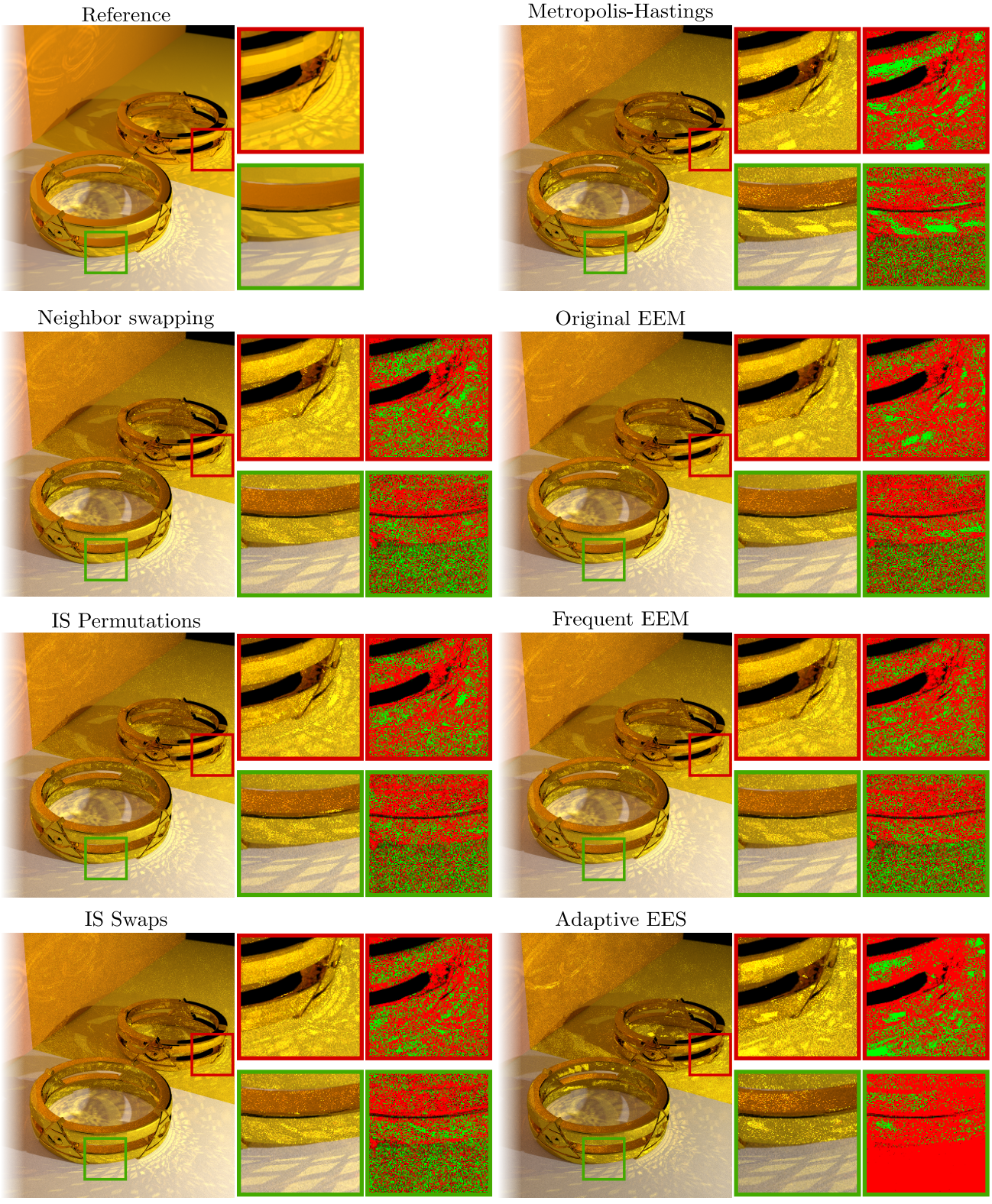

Figure 5.14: Equal-time (15 minutes) comparison of the different replica exchange moves and the adaptive equi-energy sampler in light transport simulation. Note that all the methods here are built on top of primary sample space MLT [Kelemen et al., 2002] with additional regularization, see Section 5.4. To better distinguish which method has better global exploration, one can observe how uniform is the noise in the corresponding positive (green) - negative (red) difference images.

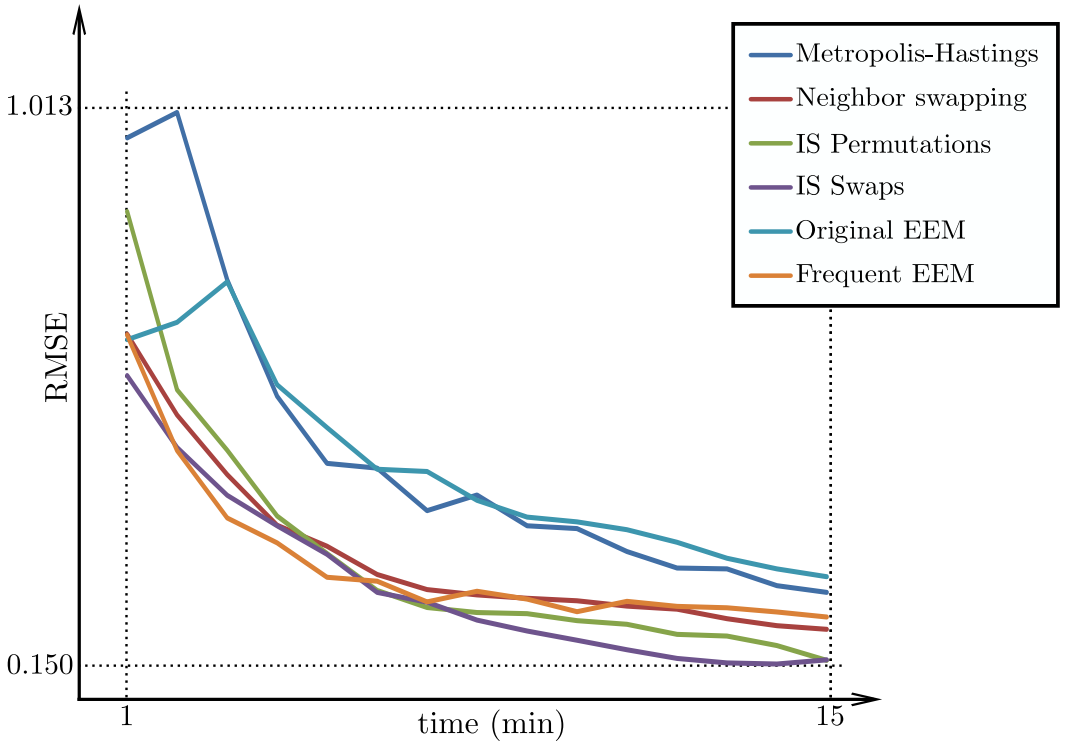

Figure 5.15: Here we show convergence plots of the different replica exchange moves in light transport simulation. Note that we exclude from this figure the adaptive equi-energy sampler (which has the lowest value of RMSE equal to 2.440) to better show the differences among the other methods. For the corresponding visual results of these methods see Figure 5.14.

over-samples or under-samples some parts of the path space. Our method built on top of PSSMLT has better global exploration, however it often cannot sufficiently exploit the discovered paths. This different behavior is caused by the fact that PSSMLT maps one sample from the primary sample space to a whole tree of different paths. This leads to a flatter target function and thus better global exploration. On the other hand MMLT maps the primary sample to just one path, which allows it to sample a mapping that leads to better local exploration of this path. However, the resulting target function has higher variation and thus hinders global exploration.

Chains and temperatures. By trial and error we have determined that using 8 chains is most effective in our test scenes. The chains use target functions with temperatures T_i that span from 1 to 2500 and are evenly spaced on a logarithmic scale (i.e. T_i/T_{i+1} is constant for all i), which corresponds to our synthetic tests in Section 5.3.7. Note that the highest temperature $T_8 = 2500$ was chosen so that the corresponding chain could easily explore the tempered path space. For a chain c_i with the temperature $T_i > 1$ we increase the original roughness $\rho \geq 0$ of BSDFs at connecting vertices using Equation (5.5). Note that MMLT uses one chain per path length and thus after applying replica exchange with 8 different temperatures, the number of chains used in the MMLT version of our algorithm is $8k$, where k is the maximum length of a path.

Regularization. Our algorithm will work as is for highly glossy bidirectional scattering distribution functions, however it will fail to swap any path that has

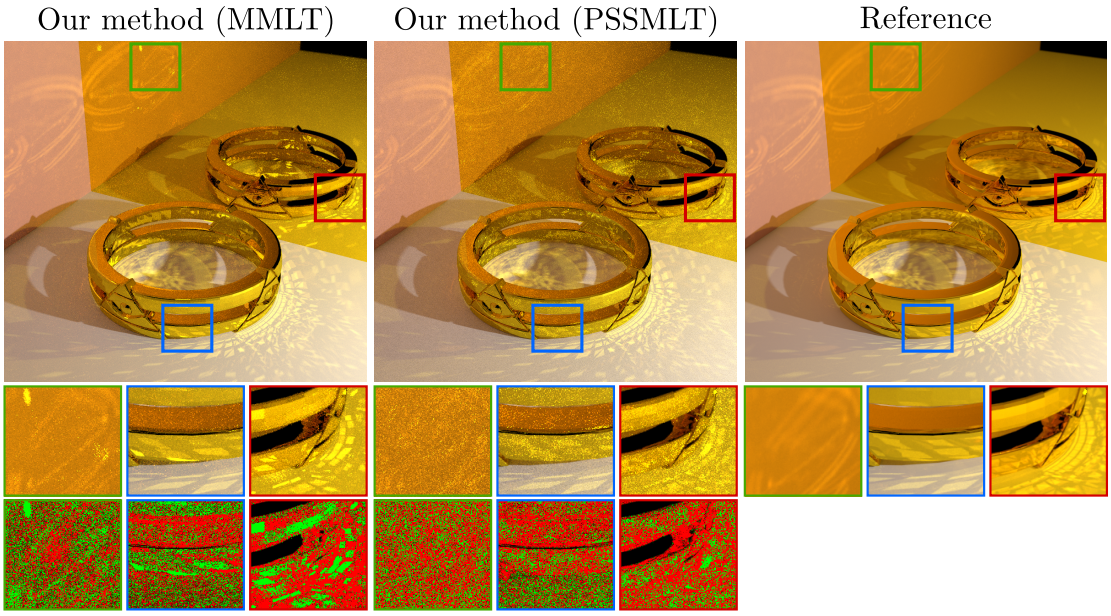

Figure 5.16: Equal-time (15 minutes) comparison of our method built on top of multiplexed MLT (MMLT) and primary sample space MLT (PSSMLT). While our method built on top of MMLT delivers cleaner result due to better local exploration, it over-samples or under-samples some parts of the path space (which exhibits as non-uniform noise in the corresponding positive (green) - negative (red) difference images). Our method built on top of PSSMLT has better global exploration, however, it often cannot sufficiently exploit the discovered paths.

specular or near-specular BSDFs at connecting vertices. Let us consider a path $\overline{x}$ that has specular BSDFs at connecting vertices and has non-zero contribution in the untempered light transport simulation. Such a path cannot be discovered by any chain with higher temperature (for near-specular BSDFs the chance of discovering $\overline{x}$ will be extremely low). While any chain with higher temperature will generate many similar paths that have zero contribution in the untempered light transport simulation, it will have zero-probability of generating exactly the same path $\overline{x}$. To solve this issue we regularize light transport simulation [Kaplanyan and Dachsbacher, 2013] by further widening the lobes of specular/near-specular BSDFs at the connecting vertices. This allows us to swap to the first chain paths that are just similar to the path $\overline{x}$ and thus our algorithm can efficiently explore the state space even in the presence of specular/near-specular BSDFs.

More specifically, we borrow a formula used by Kaplanyan and Dachsbacher [2013] and we use it to increase the roughness of BSDFs (assuming the Beckmann distribution) at the connecting vertices to be at least $\rho_0\gamma^n$, where the initial roughness ρ_0 and the regularization decline rate γ are user parameters and n is the number of the current sample generated by a Markov chain. With an increasing number of generated samples the roughness of BSDFs will thus converge to its original value. Note that we use this additional regularization for all chains in our algorithm and we apply it prior to tempering.

Mutations. We apply the same small step and large step mutations as in PSSMLT (see Section 4.2.1), while settings the probability of the large step mutation to 0.3. Note that we have experimented with using adaptive muta-

tions as described in Section 4.4, but it has not improved the performance of our algorithm.

Chain initialization. To initialize each chain we apply the start-up bias elimination method described in Section 3.6. To generate initial samples for chains with temperature $T > 1$, we generate the samples using bidirectional path tracing in the tempered light transport simulation. In order to ensure correct normalization when utilizing regularization, we update the normalization constants using independent samples created by large step mutations [Kaplanyan and Dachsbacher, 2013].

Combining contribution from all chains. Using only samples from the chain with the original target function to render the image would be wasteful as the majority of the samples would be thrown away. We therefore combine the contributions of all samples generated by our chains using multiple importance sampling (MIS, see Section 2.3) weights. The MCMC MIS weight for the given sample $\overline{u}_i$ from the chain c_i is defined as

$$w_{\text{mcmc}}(c_i, \overline{u}_i) = \frac{\frac{\pi_i(\overline{u}_i)}{b_{c_i}}}{\sum_{j=1}^{M} \frac{\pi_j(\overline{u}_i)}{b_{c_j}}}, \tag{5.13}$$

where b_{c_j} is the normalization constant of the chain c_j computed from the initial samples. Figure 5.17 shows the contribution of each chain to the final image. Kelemen et al. [2002] use a similar approach, where they apply MIS to combine samples from a MCMC sampler and independently generated samples (i.e. the large steps).

5.5 Results

We have implemented both versions of our algorithm on top of the Mitsuba renderer [Jakob, 2010]. We ran all the experiments on a PC with an Intel Core i7 at 3.50 GHz with 32 GB RAM using eight logical cores. All the reference images were rendered using the VCM/UPS algorithm (see Section 2.7).

More specifically, we compare our algorithm with primary sample space MLT (PSSMLT, see Section 4.2.1), with manifold exploration light transport (MELT, see Section 4.3.1), and with multiplexed MLT (MMLT, see Section 4.2.2). The PSSMLT and MMLT algorithms use the same mutations as we do (see Section 5.4), while MELT's parameters used in Mitsuba have been manually tuned for each scene to ensure the best performance.

All the tested algorithms including ours use additional path space regularization that allows sampling of otherwise hard-to-sample paths. This regularization is implemented by widening the BSDF lobes at connecting vertices, similar to the tempering of light transport simulation (see Section 5.4). For primary sample space algorithms we set the initial roughness ρ_0 to 4×10^{-3} for all our test scenes and tune the regularization decline rate γ for each scene separately ($\gamma = 10^{-4}$ is used for the **Ring** scene, while $\gamma = 10^{-5}$ is used for other scenes). For MELT we set $\rho_0 = 1 \times 10^{-2}$ and $\gamma = 10^{-5}$ for all scenes, since it allowed the algorithm to better explore the state space. Note that without this regularization, none of the algorithms would be able to properly explore the state space in the presence of hard-to-sample paths (see Figure 5.2).

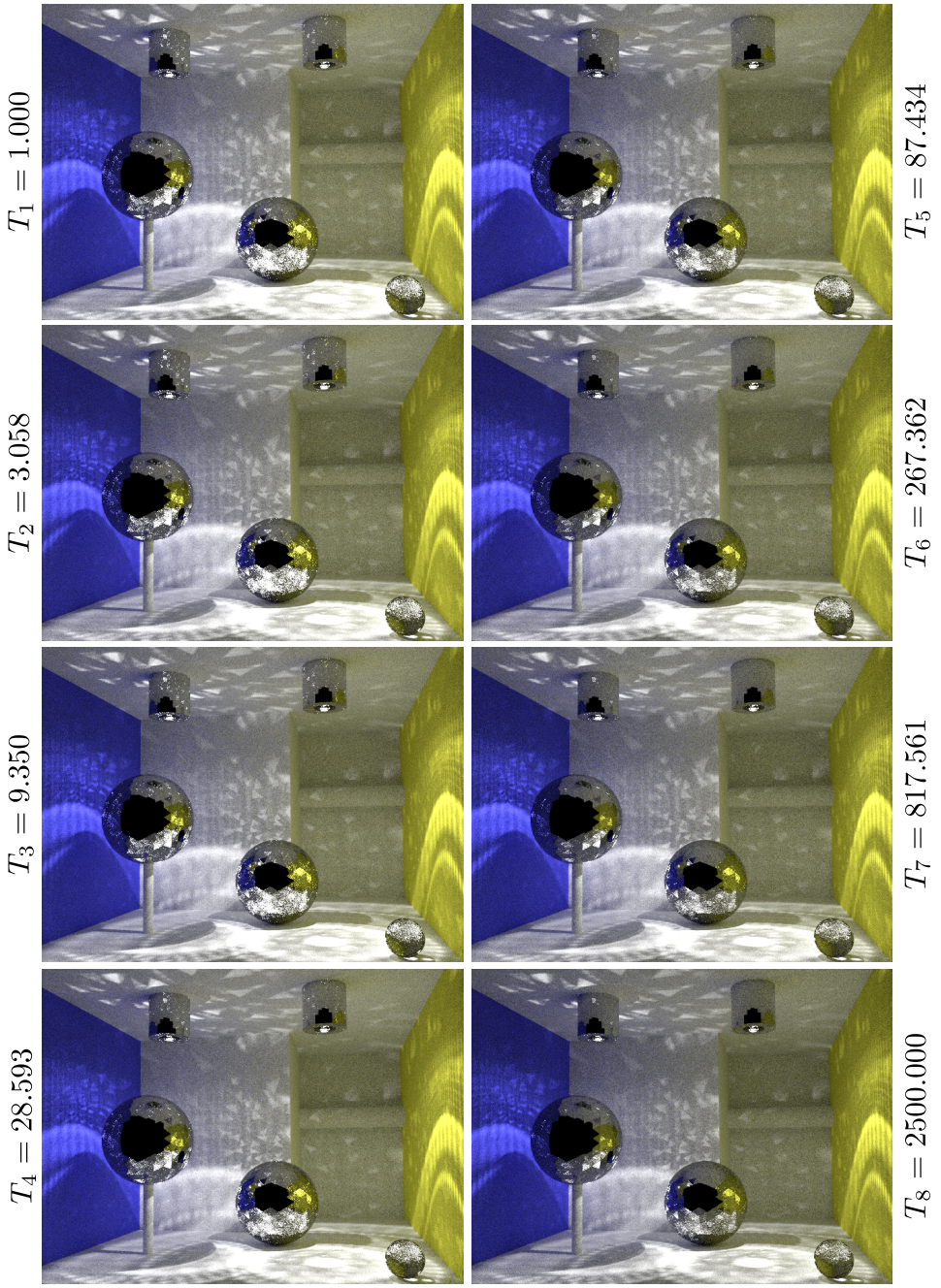

Figure 5.17: The contribution of each chain to the final image weighted by multiple importance sampling weights.

Note that in the following visual comparisons we also show the positive (green) - negative (red) difference images to better demonstrate which method has better global exploration. A method with superior global exploration should exhibit uniform green-red noise in the difference image.

5.5.1 The PSSMLT variant

Figures 5.1, 5.18, 5.19 show equal-time comparisons of our algorithm built on top of PSSMLT, regularized PSSMLT, and regularized MELT. All three cases represent a typical scene configuration featuring hard-to-sample paths (e.g. specular-

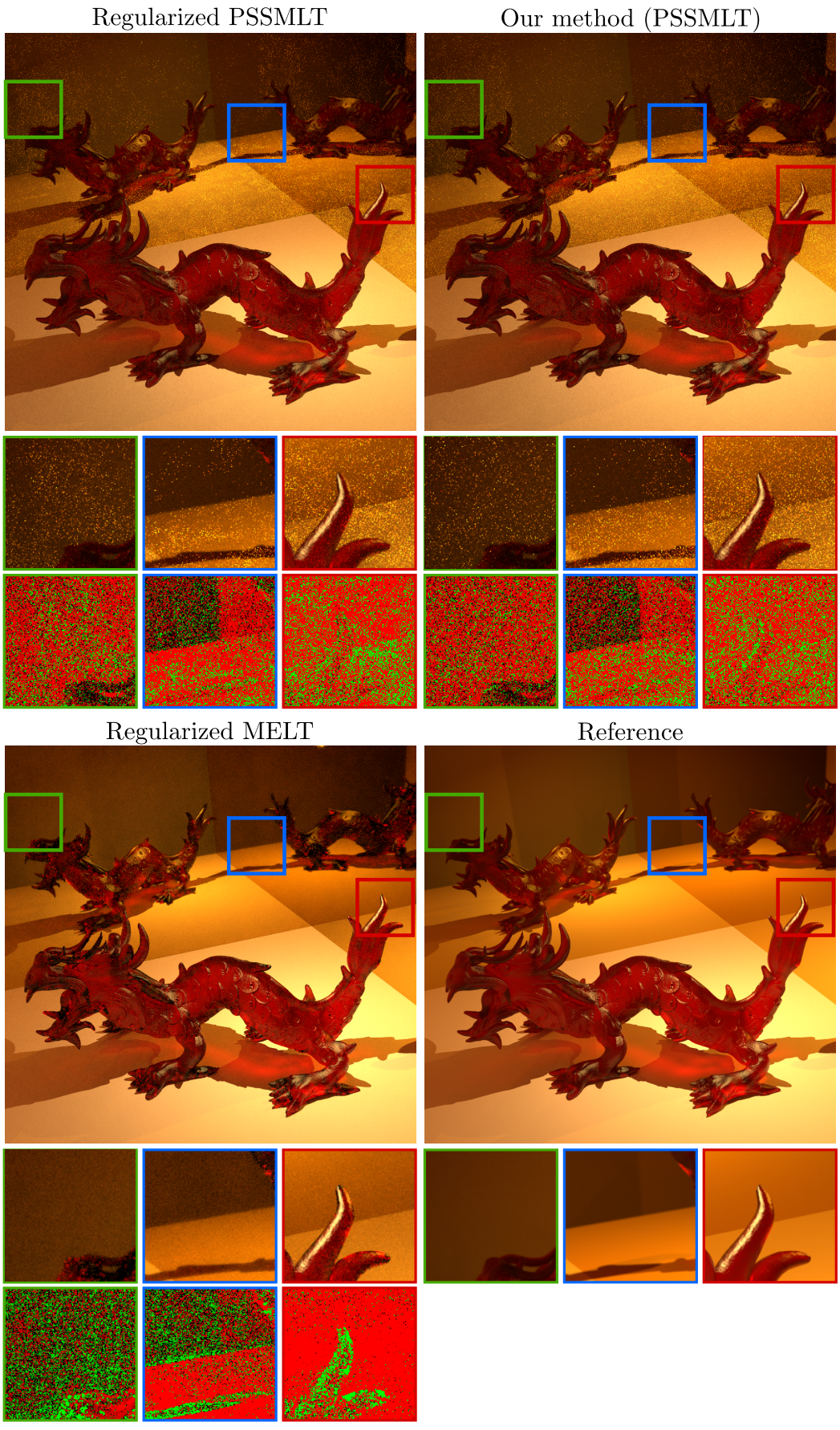

Figure 5.18: An equal-time comparison (15 minutes) of our method built on top of PSSMLT in the *Dragon* scene with PSSMLT (see Section 4.2.1) and manifold exploration light transport (MELT, see Section 4.3.1), both with additional regularization (see Section 4.5.4).

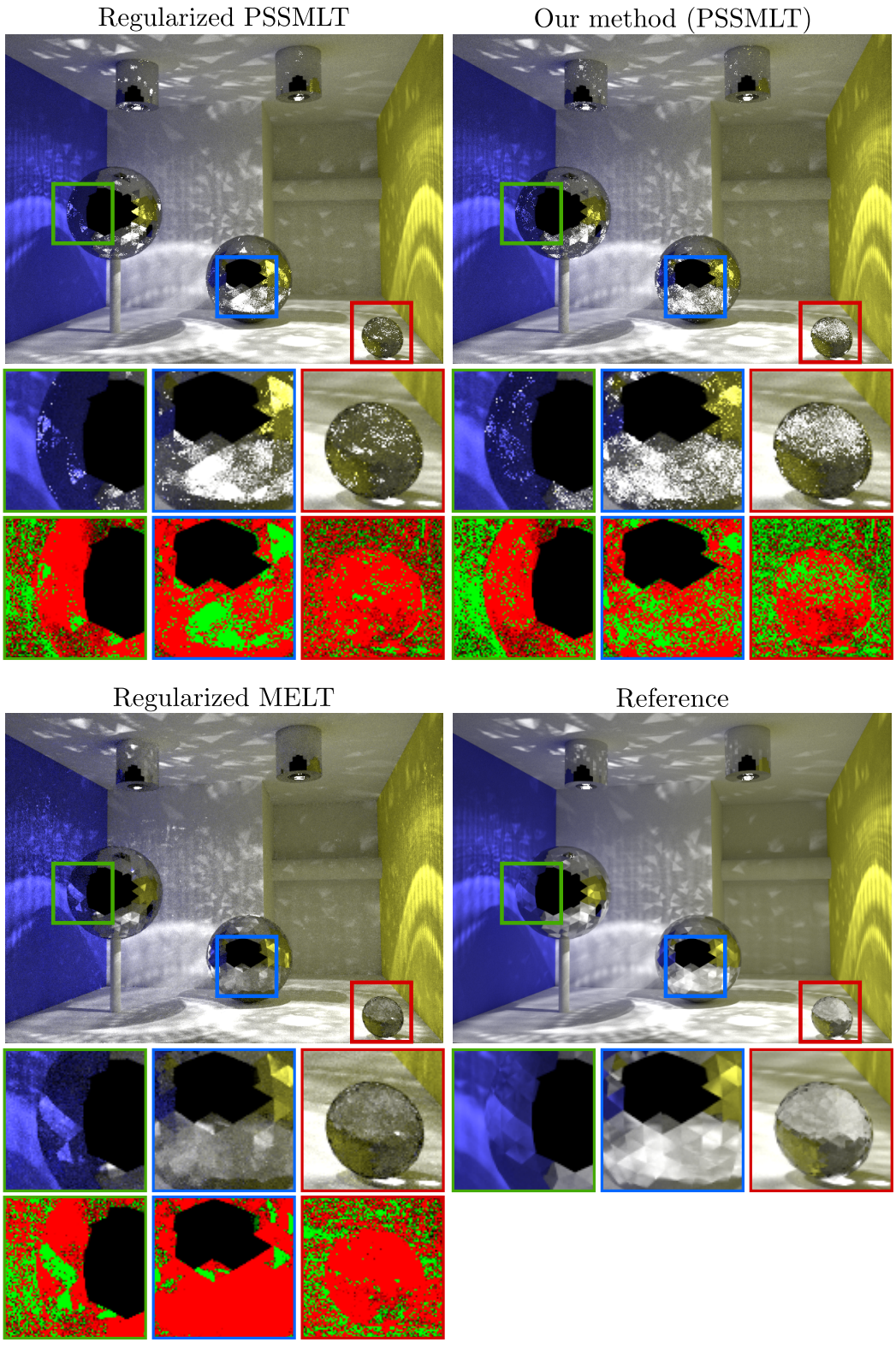

Figure 5.19: An equal-time comparison (15 minutes) of our method built on top of PSSMLT in the *Mirror-balls* scene with PSSMLT (see Section 4.2.1) and manifold exploration light transport (MELT, see Section 4.3.1), both with additional regularization (see Section 4.5.4).

	Ring	**Dragon**	**Mirror-balls**
Image resolution	1024×1024	1024×1024	640×480
#mutations PSSMLT	118.28	61.92	340.07
#mutations MMLT	1077.16	N/A	2504.76
#mutations our algorithm (PSSMLT)	107.90	60.55	307.62
#mutations our algorithm (MMLT)	710.72	N/A	1617.12
#mutations MELT	1043.04	959.02	2616.31

Table 5.4: Settings and statistics for scenes in Figures 5.1, 5.18, 5.19, 5.21. Note that the number of mutations is reported as an average across image pixels.

diffuse-specular paths), where global exploration in the current MCMC methods is insufficient. Table 5.4 summarizes rendering parameters and statistics.

While the results generated by regularized MELT have the lowest amount of noise due to its superior local exploration mutations, global exploration in MELT is insufficient and thus MELT fails to properly sample all important paths. Even in the **Dragon** scene it fails to properly sample a huge reflected caustic observable in the back wall mirror. In the **Mirror-balls** scene it seems to find most of the hard-to-sample paths, however it cannot regularly visit them and thus they remain under-sampled. This fact is directly observable from the positive-negative difference images.

Regularized PSSMLT is certainly better at exploring the whole state space compared to regularized MELT, but its global exploration is still insufficient. In the **Ring** scene it fails to find all the small reflected caustics. Other times it does not sample the important paths uniformly, which leads to splotchy patterns, as observable in the **Dragon** scene.

Our algorithm achieves the best global exploration from the tested algorithms, which can be observed from the more uniform noise in positive-negative difference images. However, as regularized PSSMLT, it suffers from insufficient local exploration and thus the resulting image has more noise compared to MELT's result. While our algorithm certainly improves global exploration compared to PSSMLT and MELT, it does not converge completely uniformly and thus at first it might miss important paths. However, it will over time converge to the true solution as shown in Figure 5.20.

5.5.2 The MMLT variant

As previously shown in Figure 5.16, our algorithm built on top of MMLT has better local exploration than the variant built on top of the PSSMLT. On the other hand, it has worse global exploration and thus it only slowly discovers all the hard-to-sample paths.

However, even our algorithm built on top of MMLT is able to improve global exploration compared to the regularized MMLT as shown in Figure 5.21. In both the **Ring** scene and the **Mirror-balls** scene, our algorithm tends to discover more important parts of the path space compared to regularized MMLT. On the other hand, our algorithm built on top of MMLT does not sample the important paths regularly, which can be observed from the appearance of correlation artifacts and from the difference images (especially in the **Ring** scene). This in turn leads to

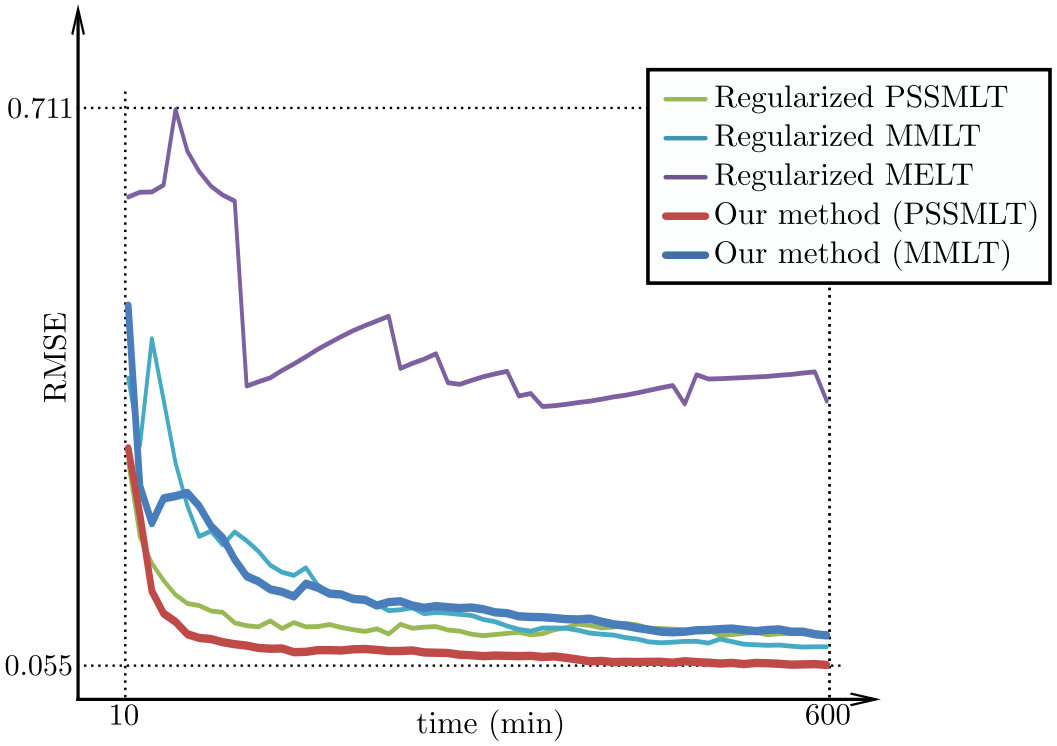

Figure 5.20: Here we show convergence plots in the **Ring** scene of primary sample space MLT (PSSMLT, see Section 4.2.1), manifold exploration light transport (MELT, see Section 4.3.1), multiplexed MLT (MMLT, see Section 4.2.2), and both variants of our algorithm. Note that all the algorithms utilize path space regularization (see Section 4.5.4). Our algorithm built on top of PSSMLT has the smoothest convergence plot from all the tested algorithms. The spikes appearing in convergence plots of other algorithms indicate either appearance of artifacts or a random discovery of important light transport paths.

convergence rate that is in longer rendering comparable to the regularized MMLT as shown in Figure 5.20.

5.6 Limitations

Tempering visibility. As discussed in Section 5.2.1 we currently cannot effectively temper visibility. For this reason, our tempering approach may not be sufficient at improving global exploration in the MCMC algorithm, especially in scenes with complex visibility. We shortly experimented with replacing the binary visibility with a function that returns non-zero values for high-temperature chains even if the connection is blocked. Unfortunately, such a simple approach did not bring any improvement. Thus to effectively temper visibility to further improve global exploration, one would have to apply a more sophisticated approach, most probably based on a spatial filtering of the visibility function.

Measured bidirectional scattering distribution functions. We achieve the tempering of light transport simulation by roughening BSDFs at connecting vertices, however not all BSDFs used in practical rendering systems have a roughness parameter. One example is a BSDF that is measured from a real-life material and

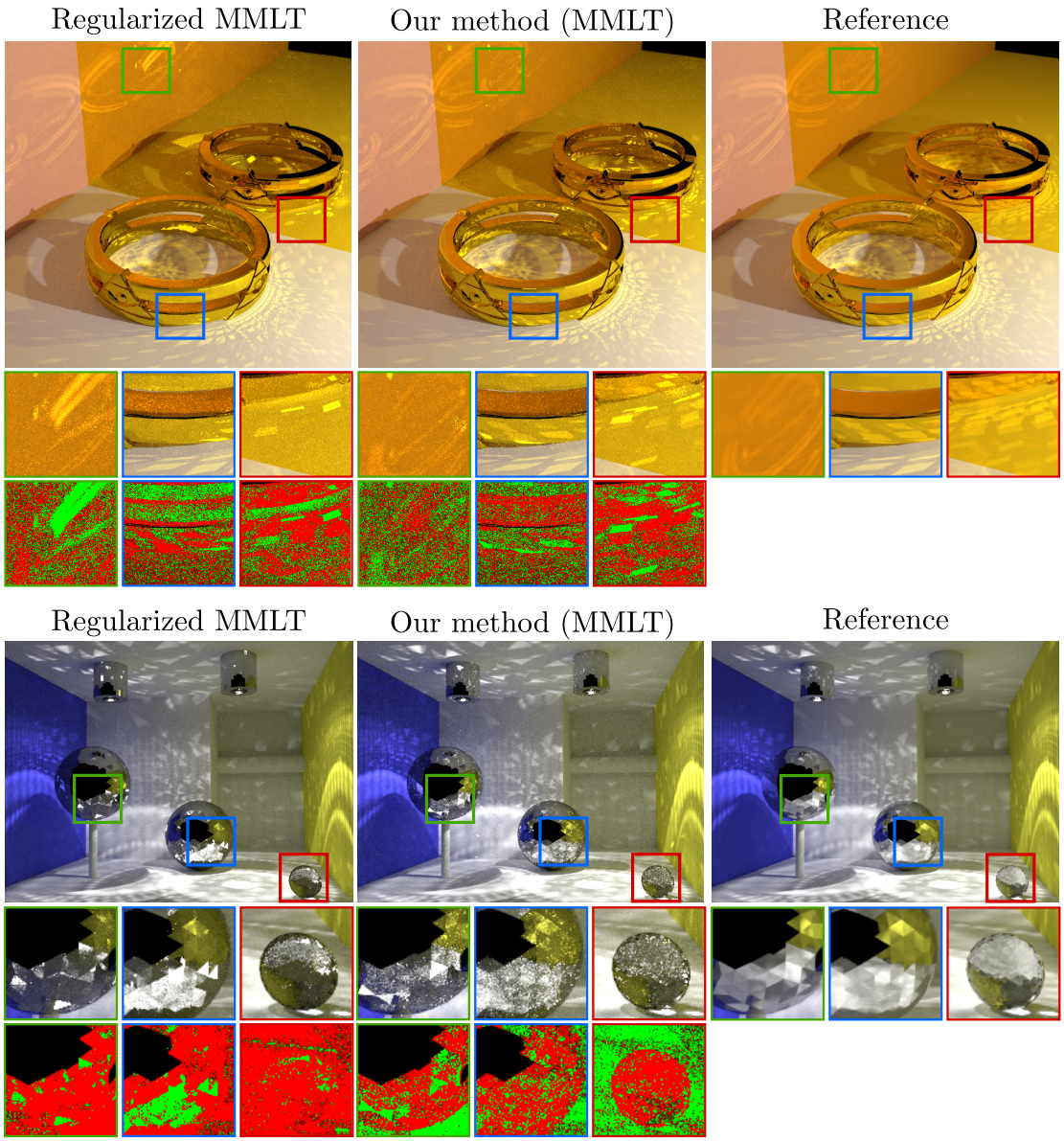

Figure 5.21: An equal-time comparison (15 minutes) of our method built on top of MMLT in the *Ring* scene and *Mirrorballs* scene with MMLT (see Section 4.2.2) with additional regularization (see Section 4.5.4).

stored in a tabulated form. To enable efficient tempering in such a case, we could fit a parametric representation to the measured data. The roughening could then be applied on the parametric representation. Note that the BSDF approximation would not need to be precise, since it would be used only for the tempered light transport simulation.

Progressive rendering. Applying replica exchange as in our algorithm certainly improves global exploration when compared to the current MCMC methods and it allows our algorithm to discover, over time, all the important paths. However, when we observe the progressive results of our algorithm, we can see that some features appear only after several iterations. Figure 5.22 illustrates that behavior. This fact hinders the ability to render fast previews. In the next chapter, we discuss an MCMC algorithm that is more suitable for such a task.

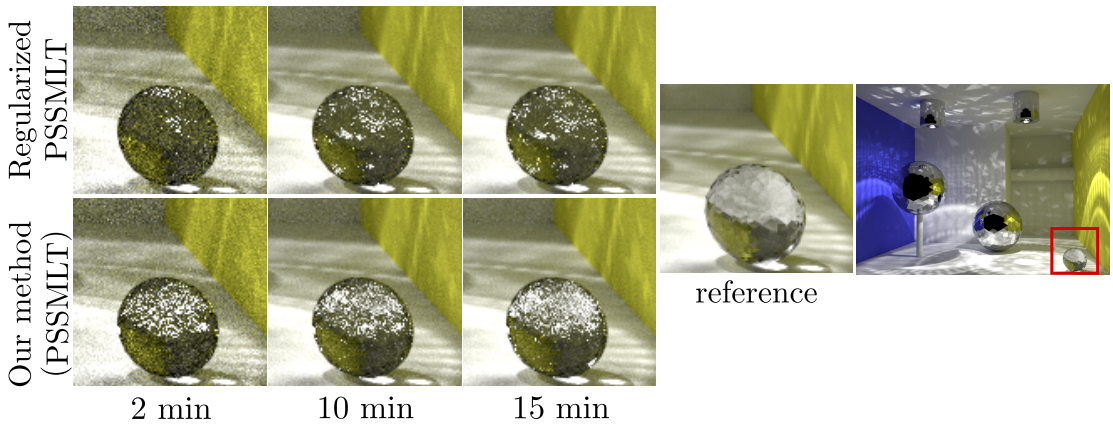

Figure 5.22: Visualization of rendering of regularized PSSMLT with and without our method in different time steps. While our algorithm certainly improves global exploration, we can see that it still discovers the different important parts of the path space in a random order and thus is unsuitable for rendering fast previews.

5.7 Future work

While replica exchange has proven to be a good tool for improving global exploration, from both the high-dimensional synthetic test (see Section 5.3.7) and the rendering results (see Section 5.5), it seems that this approach alone cannot completely solve the problem with insufficient global exploration. In this section we shortly discuss two other techniques used in general MCMC literature, that could potentially further improve global exploration.

Delayed rejection. *Delayed rejection* [Mira, 2001] is a technique that allows to accept another proposal when the previous one was rejected (without repeating the current sample). This allows us to design an algorithm that creates the first proposal using a mutation that generates samples farther from the current one and only if this proposal is not accepted another proposal can be created using a more local mutation. Since the current sample is not repeated when the first proposal is rejected, the correlation of samples is reduced compared to original Metropolis-Hastings. Generating proposals in light transport is expensive and thus one might consider combining this technique with replica exchange. We can first try a cheap replica exchange move and if that fails, we can mutate the chain using a local mutation instead. Note that delayed rejection can be iterated and thus we could try to swap with all chains before resorting to a local mutation.

Multiple-try Metropolis. *Multiple-try Metropolis* is a technique that allows us to pick the best candidate from several proposals and then accept this candidate with higher probability than in usual Metropolis-Hastings. Section 4.6 discusses this technique in more detail. Since the technique requires generating several proposals for each accepted sample, it can be quite expensive and thus is currently mainly used in methods where expensive generation of samples is not a limitation (see Section 4.6.1). However, we believe this technique could improve global exploration if applied selectively in cases where the chain gets stuck at a mode of the target function. Note that an algorithm that applies multiple-try Metropolis in such a fashion would require a rigorous proof that it does not break MCMC convergence. Furthermore, we could combine the algorithm with replica exchange

by using several different replica exchange moves as the proposals of multiple-try Metropolis.

5.8 Conclusion

In this chapter we focused on effectively applying replica exchange to improve global exploration in MCMC algorithms in light transport simulation. To achieve that goal we directly tailored tempering for light transport simulation. This allowed us to improve exploration of hard-to-sample paths in the scene. We discussed several different replica exchange moves. Based on the inefficiency of the current methods we proposed novel ones: importance-sampled permutations and swaps, and frequent equi-energy moves. We compared the different moves with the equi-energy sampling algorithm [Kou et al., 2006] in both synthetic and light transport tests to determine the most robust option. The proposed tempering for light transport simulation together with the selected replica exchange algorithm were combined into a novel MCMC light transport algorithm. The presented results demonstrate that our algorithm indeed in many cases improves global exploration compared to the existing MCMC algorithms in the presence of difficult light transport. However, we acknowledge that this is just the first step in achieving sufficient global exploration in all cases and we thus pointed out some possible directions for future work.

Appendix

5.A Other tempering approaches

In this appendix we discuss how we have arrived to the final tempering of light transport simulation that we use in our algorithm, see Section 5.2. We briefly describe here other attempts at tempering light transport simulation that we discovered to be ineffective.

5.A.1 Standard tempering

In general Markov chain Monte Carlo literature [Earl and Deem, 2005], the tempering of the simulation is usually approached by raising the original target function π to the power of the inverse temperature as discussed in Section 4.5.2. Since this technique can be also directly applied in light transport simulation, it was the first tempering we used in our research. In the following text, we show how we applied standard tempering in both the path space and the primary sample space algorithms and we explain why such tempering turned out to be inefficient in light transport simulation.

Standard tempering in the path space

Let us first consider the MCMC light transport algorithms that operate directly in the path space (e.g. the original Metropolis light transport algorithm, see Section 4.1). In this case, the target function is proportional to the contribution function f of the proposed light transport path. The density of the i-th chain with temperature T_i is defined as

$$\pi_i(\overline{x}) = f(\overline{x})^{(1/T_i)}. \tag{5.14}$$

In order to apply such tempering, we, however, had to first solve the following two issues.

Numerical instability. The computation of the tempered target function π_i is very numerically unstable. This is caused by the contribution function f itself (see Section 2.1 for its definition), since it can reach both low and high values that cannot be precisely represented by floating point numbers (even with double precision). This is not an issue in the standard Metropolis-Hastings algorithm. When one computes the Metropolis-Hastings acceptance ratio (Equation (3.2)) in algorithms that operate in the path space, one computes the term $\frac{\pi(v)}{q(u \to v)}$ which is more numerically stable than computing $\pi(v)$ itself and the mutation probability $q(u \to v)$ separately. Yet this becomes a problem when we want to compute the swap ratio in replica exchange

$$\frac{\pi_i(\overline{x}_j)\pi_j(\overline{x}_i)}{\pi_i(\overline{x}_i)\pi_j(\overline{x}_j)}, \tag{5.15}$$

where $\overline{x}_i$ is the current state of the chain c_i (similarly for the chain c_j).

A way to overcome this problem is to compute the logarithm of the target function π_i and then modify the swap ratio accordingly

$$\begin{aligned}\frac{\pi_i(\overline{x}_j)\pi_j(\overline{x}_i)}{\pi_i(\overline{x}_i)\pi_j(\overline{x}_j)} &= \frac{f(\overline{x}_j)^{(1/T_i)}f(\overline{x}_i)^{(1/T_j)}}{f(\overline{x}_i)^{(1/T_i)}f(\overline{x}_j)^{(1/T_j)}} \\ &= \frac{e^{(1/T_i)\ln f(\overline{x}_j)+(1/T_j)\ln f(\overline{x}_i)}}{e^{(1/T_i)\ln f(\overline{x}_i)+(1/T_j)\ln f(\overline{x}_j)}} \\ &= e^{(1/T_i-1/T_j)(\ln f(\overline{x}_j)-\ln f(\overline{x}_i))}. \end{aligned} \tag{5.16}$$

Note that computing the logarithm of the contribution function $\ln f(\overline{x})$ is straightforward and numerically stable, since f is a product of many simpler terms and thus $\ln f(\overline{x})$ is a sum of logarithms of these terms. Another approach is to directly compute the ratio $\frac{\pi_j(\overline{x}_i)}{\pi_i(\overline{x}_i)}$, again utilizing the fact that the contribution function f is a product of simpler terms.

Spanned distance problem. The contribution function f in the path space Ω contains a product of geometric factors between each pair of neighboring vertices of the path $\overline{x}$ (see Equation (2.2)). Each geometric factor $G(x_i \leftrightarrow x_{i+1})$ is a product of terms, where one of them is an inverse squared distance between the two vertices x_i and x_{i+1} (see Equation (2.3)). This means that a path that spans long distance in the scene will have a much lower contribution function than a similar path that spans shorter distance. While tempering the contribution function reduces the differences, the swapping ratio between two chains that have current paths with very different spanned distances is still very low. Second problem is that the tempering is not invariant to scene scaling. To solve these problems, we sacrificed the flatness of the tempered target function and left the squared distance product untempered

$$\pi_i(\overline{x}) = \left(\frac{f(\overline{x})}{\prod_{i=0}^{k-1}\frac{1}{|x_i-x_{i+1}|^2}}\right)^{(1/T_i)} \prod_{i=0}^{k-1}\frac{1}{|x_i-x_{i+1}|^2}. \tag{5.17}$$

Standard tempering in the primary sample space

The two above issues are not present in the primary sample space (see Section 4.2). There the target function contains the contribution function divided by the mapping probability, which corresponds to the probability of sampling a path using a given path sampling technique (see Section 4.2.1). This significantly reduces the variation of the target function, so no numerical instabilities occur. It also removes all of the geometric factors except between two connected vertices. To ensure invariance to scene scaling we therefore did not temper the inverse square distance between the connected vertices. Otherwise the application of standard tempering in the primary sample space was straightforward.

Issue of bidirectional scattering distribution functions

At this point of our research we could apply standard tempering in both the path space and the primary sample space. Figure 5.23 illustrates the standard tempering in the manifold exploration light transport algorithm (see Section 4.3.1) and in the primary sample space MLT (PSSMLT) algorithm. We can see that the

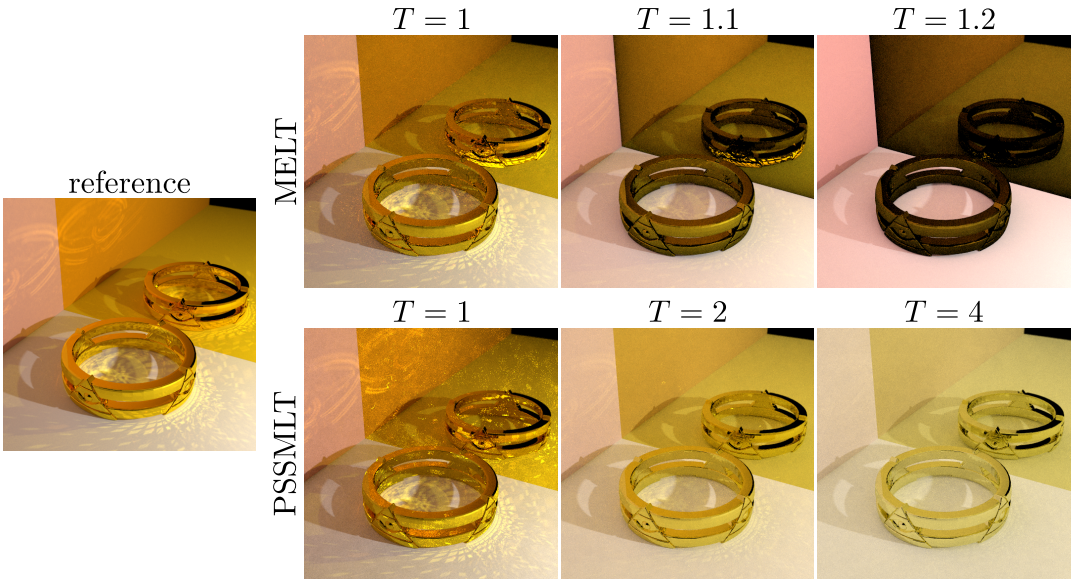

Figure 5.23: Result of applying standard tempering in the path space, rendered by MELT (see Section 4.3.1), and in the primary sample space, rendered by PSSMLT (see Section 4.2.1). We can observe that increasing temperature T leads to vanishing of caustics as discussed in the main text. Note that all specular/highly glossy BSDFs that are not importance sampled are toned down. Thus in MELT, where no BSDF is importance-sampled, all the paths except for completely diffuse ones are much darker in the tempered path space.

caustics that were present in the original untempered transport fade away in the tempered version. Furthermore, the reflected caustics that were not sampled by the original target function are not sampled in the tempered simulation either. This issue is caused by the fact that the peaks of the BSDFs are toned down in the tempered path space instead of being widened to allow their easier exploration. If we consider a beam of light bouncing off a surface with a nearly specular BSDF, most of its energy will be in the peak, while the rest of the hemisphere will have such a low energy that it can be considered zero in the floating point arithmetic (and is in fact zero for perfectly specular BSDF). Thus tempering the contribution function as a whole without taking a special care of BSDFs will only reduce the peak of a BSDF instead of widening it as illustrated in Figure 5.24. This in turn means that the hard-to-sample paths were not properly explored when we used this tempering of light transport simulation.

5.A.2 Tempering by roughening all BSDFs

Since applying standard tempering directly has proven to be ineffective, we instead attempted to temper the light transport simulation by directly roughening all the BSDFs in a scene (not just at the connecting vertices as we do in the final solution, see Section 5.2.1). Figure 5.25 shows a scene rendered with PSSMLT while all BSDFs are continuously tempered using the roughness parameter. We see that increasing the roughness of all BSDFs indeed allows the MCMC algorithm to more easily explore the state space. However, roughening of all BSDFs leads to a very different target function than the original one (both in the path space

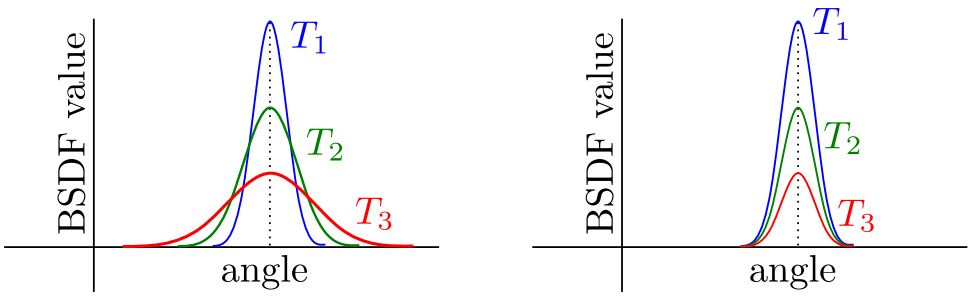

Figure 5.24: (a) When we temper the contribution function by raising its power to the inverse of temperature $1 = T_1 < T_2 < T_3$, the BSDF lobe should be widened to allow easy exploration of hard-to-sample paths. (b) However, due to floating point arithmetic inaccuracy (or the fact that the original BSDF has a delta distribution) the BSDF value is toned down and the peak is not widened.

and the primary sample space) which results in low probability of swaps between two chains with different amount of roughening of the BSDFs. Decreasing the roughening amount allows more swaps, but it also reduces the capability of ex-

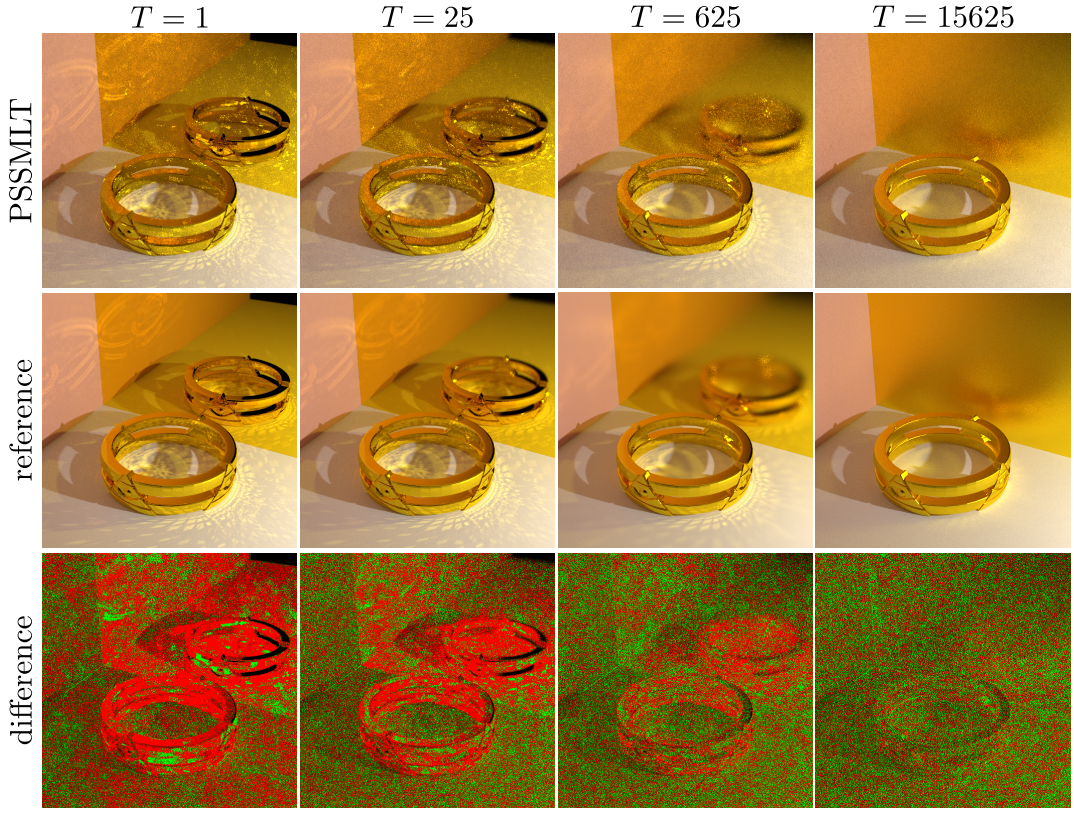

Figure 5.25: Renderings of the same scene with increasing roughness of all BSDFs (original roughness of the microfacet BSDF with Beckmann distribution [Ward, 1992] is modified according to Equation (5.5)). Primary sample space MLT (PSSMLT, see Section 4.2.2) can more easily explore the state space with increasing roughness, which is exhibited by the increasingly more uniform noise in the positive (green) - negative (red) difference images.

ploring the whole state space. We tried to solve this issue by using a high number of chains, but in such a case the information from the last chain almost never reached the first chain with the original target function. Furthermore, computational cost of calculating swap probabilities was very high, since we needed to reevaluate the BSDF value of each vertex of the two paths that were being swapped. Therefore, we have abandoned this attempt as well. Instead we only tempered the BSDF at the connecting vertices, which modifies the simulation less and thus allows higher swap ratios.

5.A.3 Tempering of the connection BSDFs

First we considered the tempering of the connection BSDFs in the path space MCMC methods, e.g. the original MLT. The mutations of the path space usually involve connection of two subpaths (e.g. mutations shown in Figure 4.1) and thus we have roughened BSDFs at the connected vertices of the subpaths. However, we discovered that such an approach is less effective than when we apply a similar approach in the primary sample space.

In the primary sample space the paths are usually constructed using path sampling techniques from bidirectional path tracing (see Section 2.5) and thus they also involve connection and we can temper the BSDFs at the connecting vertices. The advantage here is that the primary sample space target function becomes flatter when tempering the connection BSDFs than the one used in the path space methods thanks to the inherent importance sampling of the two connected subpaths. Therefore we have chosen to use in our method tempering in the primary sample space.

5.B Correctness of replica exchange moves

In Section 5.3 we have presented several novel replica exchange moves, specifically importance-sampled permutations and swaps, and frequent equi-energy moves. In this section we prove that applying these replica exchange moves does not break the detailed balance condition (Equation (3.8)) and thus in turn the resulting MCMC algorithm converges to the target distribution. The other conditions for MCMC convergence – aperiodicity and ergodicity – are satisfied as in the original Metropolis-Hastings algorithm, see Section 3.4.

Instead of directly proving that detailed balance holds, we can realize that a swap of states between two chains is just a mutation that happens on a product chain in a product state space. Given M Markov chains in the state space $\mathcal{U}$ with the target functions $\pi_1, \ldots, \pi_M$ and the current states $u_1, \ldots, u_M$, we can define a product chain that operates in $\overline{\mathcal{U}} = \mathcal{U}^M = \mathcal{U} \times \ldots \times \mathcal{U}$ with the target function $\overline{\pi}(\overline{u}) = \prod_{i=1}^{M} \pi_i(u_i)$. When the product chain converges to the desired product distribution given by the target function $\overline{\pi}(\overline{u})$, each chain c_i converges to the distribution given by π_i as well [Geyer, 1991]. The Metropolis-Hastings acceptance ratio of the product chain is defined as

$$\alpha(\overline{u} \to \overline{v}) = \min\left(\frac{\overline{\pi}(\overline{v})q(\overline{v} \to \overline{u})}{\overline{\pi}(\overline{u})q(\overline{u} \to \overline{v})}, 1\right) = \min\left(\frac{\left(\prod_{i=1}^{M} \pi_i(v_i)\right) q(\overline{v} \to \overline{u})}{\left(\prod_{i=1}^{M} \pi_i(u_i)\right) q(\overline{u} \to \overline{v})}, 1\right). \tag{5.18}$$

The mutation probability for the common replica exchange moves is symmetric (i.e. $q(\overline{v} \to \overline{u}) = q(\overline{u} \to \overline{v})$) and thus cancels out. Since usually only two chains exchange their states, most of the product target functions cancel out as well and we arrive to the standard swap formula (Equation (4.11)).

Thus to prove that our proposed replica exchange moves do not break detailed balance, we only need to ensure that $q(\overline{v} \to \overline{u})$ is positive if $q(\overline{u} \to \overline{v})$ is positive, otherwise the move is not feasible. In the cases where these two probabilities are not equal, we have to include them into the swap acceptance formula.

5.B.1 Importance-sampled permutations

The correctness of accepting a permutation of the current states of the chains according to Equation (5.8) is immediately visible from Equation (5.18) if we substitute $q(\overline{v} \to \overline{u}) = p(o^{-1})$ and $q(\overline{u} \to \overline{v}) = p(o)$.

If we now consider the set of permutations $\mathcal{O}$ to include all possible permutations on $\{1, \ldots, M\}$, including identity, we get

$$\begin{aligned}
\alpha(\overline{u} \to \overline{v}) &= \min\left(\frac{\prod_{i=1}^{M} \pi_i(u_{o(i)}) p(o^{-1})}{\prod_{i=1}^{M} \pi_i(u_i) p(o)}, 1\right) \\
&= \min\left(\frac{\prod_{i=1}^{M} \pi_i(u_{o(i)}) \frac{\prod_{i=1}^{M} \pi_i(u_i)}{\sum_{o' \in \mathcal{O}} \prod_{i=1}^{M} \pi_i(u_{o'(i)})}}{\prod_{i=1}^{M} \pi_i(u_i) \frac{\prod_{i=1}^{M} \pi_i(u_{o(i)})}{\sum_{o' \in \mathcal{O}} \prod_{i=1}^{M} \pi_i(u_{o'(i)})}}, 1\right) \\
&= \min\left(\frac{\prod_{i=1}^{M} \pi_i(u_{o(i)}) \pi_i(u_i)}{\prod_{i=1}^{M} \pi_i(u_i) \pi_i(u_{o(i)})}, 1\right) \\
&= 1, \qquad (5.19)
\end{aligned}$$

where we have used the definition of the permutation probability $p(o)$ in Equation (5.7). Thus the importance-sampled permutation is always accepted.

5.B.2 Importance-sampled swaps

Given a current chain c_i and an importance-sampled chain c_j, the correctness of the swap acceptance probability in Equation (5.10) can be proven using Equation (5.18) as follows

$$\begin{aligned}
\alpha(\overline{u} \to \overline{v}) &= \min\left(\frac{\pi_i(u_j) \pi_j(u_i) p'(i,j)}{\pi_i(u_i) \pi_j(u_j) p(i,j)}, 1\right) \\
&= \min\left(\frac{\pi_i(u_j) \pi_j(u_i) \frac{\pi_i(u_i) \pi_j(u_j)}{\sum_{k=1}^{M} \pi_i(u'_k) \pi_k(u'_i)}}{\pi_i(u_i) \pi_j(u_j) \frac{\pi_i(u_j) \pi_j(u_i)}{\sum_{k=1}^{M} \pi_i(u_k) \pi_k(u_i)}}, 1\right) \\
&= \min\left(\frac{\sum_{k=1}^{M} \pi_i(u_k) \pi_k(u_i)}{\sum_{k=1}^{M} \pi_i(u'_k) \pi_k(u'_i)}, 1\right) \\
&= r_{\mathrm{IS}}(i,j), \qquad (5.20)
\end{aligned}$$

where we have substituted $q(\overline{u} \to \overline{v}) = p(i,j)$ and $q(\overline{v} \to \overline{u}) = p'(i,j)$. $p(i,j)$ is the probability of sampling a chain c_j for swapping with the current chain c_i (see Equation (5.9) for its definition) and $p'(i,j)$ is the probability of the same event

after the swap. The current state of the k-th chain after the swap u'_k is defined in Equation (5.11).

5.B.3 Frequent equi-energy moves

Let us assume that a chain c_j has its current state in the energy ring D_i and this ring contains current states of M_i chains (including the chain c_j). If M_i is greater than one, then a chain that will exchange its state with the chain c_j is picked with probability $1/(M_i - 1)$. This means that in Equation 5.18 the mutation probability $q(\overline{u} \to \overline{v})$ will be equal to $1/(M_i - 1)$. Now the opposite mutation probability $q(\overline{v} \to \overline{u})$ is the same, since there is the same number of the current states in D_i after the swap. Thus for this case, the mutation probability cancels out and the swap is well behaved. Note that if M_i is equal to one, we cannot swap with any other chain.

5.C Faster sampling of permutations

In Section 5.3.2 we propose a novel replica exchange move that permutes the current states of all chains using an importance-sampled permutation. Here we discuss how to efficiently perform the importance sampling of permutations.

To importance-sample a permutation from a set of possible permutations $\mathcal{O}$ on $\{1, \ldots, M\}$ given permutation probability in Equation 5.7, we must first compute the normalization factor $b_{\mathcal{O}}$. This factor can be computed without any optimization as

$$b_{\mathcal{O}} = \sum_{o \in \mathcal{O}} \prod_{i=1}^{M} \pi_i(u_{o(i)}). \tag{5.21}$$

If $\mathcal{O}$ contains all permutations of $\{1, \ldots, M\}$, the above sum goes over $M!$ summands. To speed up the summation we first define a set of permutations $\mathcal{O}_P \subseteq \mathcal{O}$ that share a common prefix $P : \{1, \ldots, l_P\} \to \{1, \ldots, M\}$ of length l_P as

$$\mathcal{O}_P = \{o \in \mathcal{O}; o(1) = P(1) \wedge \ldots o(l_P) = P(l_P)\}. \tag{5.22}$$

For example, given $M = 4$ a set $\mathcal{O}_P$ with common prefix $P = (1, 3)$ will contain permutations $(1, 3, 2, 4)$ and $(1, 3, 4, 2)$. Now we can write the summation of probabilities of permutations in $\mathcal{O}_P$ as

$$b_{\mathcal{O}_P} = \sum_{o \in \mathcal{O}_P} \prod_{i=1}^{M} \pi_i(u_{o(i)}) = \prod_{i=1}^{l_P} \pi_i(u_{P(i)}) \overbrace{\left(\sum_{o \in \mathcal{O}_P} \prod_{i=l_P+1}^{M} \pi_i(u_{o(i)}) \right)}^{\text{A common factor } F_P}. \tag{5.23}$$

We can further realize that all permutations that have a prefix $R : \{1, \ldots, l_P\} \to \{P(1), \ldots, P(l_P)\}$ (e.g. for $P = (1, 3)$, the prefix R can be either $R = (1, 3)$ or $R = (3, 1)$) share the same common factor $F_P = F_R$. This means that we can compute F_P once and then reuse it for all permutations with prefixes R.

1: **function** ComputeCommonFactor(P)
    //P is a permutation prefix of length l_P
2: **if** $l_P = M$ **then return** 1
3: **if** F_P is already computed **then return** F_P
4: Set $F_P = 0$
5: **for each** $k \in \{1, \ldots, M\}$ such that $\forall j \in \{1, \ldots, l_P\} : P(j) \neq k$ **do**
6: Define permutation prefix P' that extends P by $P'(l_P + 1) = k$
7: F_P += $\pi_{l_P+1}(u_k) \times$ ComputeCommonFactor(P')
8: Store and **return** F_P
9: Set $b_{\mathcal{O}}$ = ComputeCommonFactor(permutation prefix P with $l_P = 0$)

Figure 5.26: Algorithm for computation of permutations probability normalization $b_{\mathcal{O}}$.

5.C.1 Computing the normalization factor

Using the above knowledge, we can modify the common algorithm that generates all possible permutations to efficiently compute $b_{\mathcal{O}}$ as illustrated in the algorithm in Figure 5.26. The algorithm starts by defining a function ComputeCommonFactor (line 1) that computes F_P for a permutation prefix P of length l_P. The function first checks whether the prefix length is equal to the chain count (i.e. $l_P = M$). In that case the common factor is trivially equal to one (line 2). Otherwise it checks whether the common factor F_P was already computed (line 3) and in that case it just returns it.

If F_P has not yet been computed, we have to compute F_P by iterating over all permutations that share the prefix P. This iteration can be optimized using recursion and thus we can only iterate over all prefixes P' of length $l_P + 1$ that extend the prefix P by $P'(l_P + 1) = k$ (lines 5 - 7). The recursive call of ComputeCommonFactor then computes $F_{P'}$ for the prefix P' (line 7). Once the factor F_P is computed, we store it and return it (line 8). Calling ComputeCommonFactor with an empty prefix of zero length (i.e. $l_P = 0$) (line 9) then gives us the normalization $b_{\mathcal{O}}$, since F_P for a prefix of zero length is equal to $b_{\mathcal{O}}$.

If we realize that there are 2^M different common factors F_P (there are 2^M subsets of $\{1, \ldots, M\}$) and that each of these factors is computed by a for-cycle (line 5) with at most M iterations, we immediately see that the algorithm runs in $O(M2^M)$ and requires $O(2^M)$ memory. Even though the algorithm is exponential, it is significantly faster than iterating over all possible permutations that runs in $O(M!) \approx O(\sqrt{M}\,(M/e)^M)$.

5.C.2 Importance sampling of permutations

Now that we can efficiently compute $b_{\mathcal{O}}$, we can reuse the stored common factors F_P to sample a permutation as illustrated in the algorithm in Figure 5.27. The algorithm starts by defining a function SampleSuffix (line 1) that, given a random number $\xi \in [0, F_P)$, importance-samples a permutation from all possible permutations that share a common prefix P of length l_P. In the function we start by checking whether the prefix length is equal to M, in which case the

```
1: function SampleSuffix(P, ξ)
       //P is a permutation prefix of length l_P
       //ξ is an uniform random number in [0, F_P)
2:     if l_P = M then return P
3:     for each k ∈ {1, ..., M} such that ∀j ∈ {1, ..., l_P} : P(j) ≠ k do
4:         Define permutation prefix P' that extends P by P'(l_P + 1) = k
5:         Set p_P' = π_{l_P+1}(u_k) F_P'
6:         if ξ > p_P' then
7:             ξ −= p_P'
8:         else return SampleSuffix(P', ξ/(π_{l_P+1}(u_k)))
9: Set o = SampleSuffix(permutation prefix P with l_P = 0, ξ b_O)
```

Figure 5.27: Algorithm for fast importance sampling of permutations.

sampled permutation is equal to the prefix P (line 2). Otherwise we have to sample from a whole set of permutations that share the common prefix P. We start by iterating over all prefixes P' of length $l_P + 1$ that extend the prefix P by $P'(l_P + 1) = k$ (lines 3 - 8). In each step we compute the marginal probability $p_{P'}$ of sampling any permutation with the prefix P'. If the random number value is higher than $p_{P'}$, we will not sample any of these permutations (line 7). Otherwise, we will sample one of the permutation with the prefix P' and we let recursion take care of it (line 8). Finally, calling the function SampleSuffix with an empty prefix of zero length (i.e. $l_P = 0$) and a random number $\xi \in [0, b_{\mathcal{O}})$ gives us the importance-sampled permutation (line 9).

Since the function calls itself recursively M times and each such call performs up to M iterations of the for-cycle (line 4) the total running time is in $O(M^2)$.

6. Robust light transport simulation via Metropolised bidirectional estimators

In this chapter we discuss a different approach to solving the problem of global exploration in MCMC algorithms in scenes featuring difficult light transport. Instead of tempering the BSDFs like in the previous Chapter 5, here we take the advantage of various bidirectional estimators and subpath reuse to allow easier global exploration in scenes with the glossy/specular transport. To be concrete, the final MCMC algorithm is built upon the vertex connection and merging (VCM) [Georgiev et al., 2012], a.k.a. unified path space sampling (UPS) [Hachisuka et al., 2012] algorithm (see Section 2.7 for more information about VCM/UPS). Unlike bidirectional path tracer – that was already combined with MCMC in primary sample space MLT (PSSMLT, see Section 4.2.1) – VCM/UPS can handle even hard-to-sample paths such as specular-diffuse-specular paths (see Figure 1.1). While it is natural to expect that a combination of VCM/UPS and MCMC should work well, it still leaves a large space for different design approaches. The first proposal of a practical and efficient combination based on a number of algorithmic considerations, is the key contribution of this chapter.

To further improve global exploration and avoid inheriting the issues of correlation and unpredictable convergence of MCMC, we apply here the replica-exchange technique, similarly to the previous Chapter 5. However, here we propose to use a mixture of just two target functions: one based solely on path visibility and the other on the path contribution.

Figure 6.1 illustrates that our algorithm is better at exploring whole state space compared to existing MCMC algorithms. Furthermore, we reduce correlation in the image space and improve image-plane stratification by decoupling the sampling of light subpaths (i.e. subpaths from a light source) and camera subpaths (i.e. subpaths from the camera), and by sampling the camera subpaths by ordinary (independent) Monte Carlo.

The results demonstrate that our algorithm is indeed robust with respect to various scene configurations, see Figure 6.2 for an example. An important practical aspect is that our algorithm has a low overhead over baseline VCM/UPS, which is often implemented in the existing rendering systems [Pixar, 1982]; as such it exhibits good performance even in relatively simple cases, as we show in Figure 6.3.

6.1 Related work

Combining bidirectional methods and MCMC. Bidirectional path tracing [Lafortune and Willems, 1993, Veach and Guibas, 1994] was already used in MCMC context in PSSMLT (see Section 4.2.1). PSSMLT uses MCMC to mutate random numbers in primary sample space that are later mapped to the path space using path sampling techniques from bidirectional path tracing. In our work, we also utilize primary sample space, however we map the random numbers using

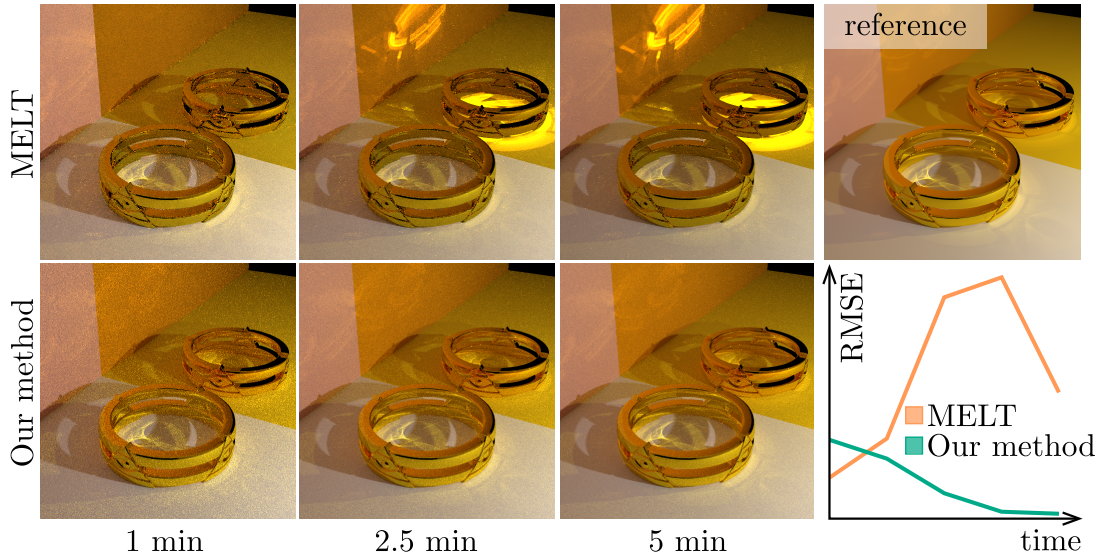

Figure 6.1: Manifold exploration light transport (MELT) [Jakob and Marschner, 2012] suffers from excessive sample correlation in scenes with specular surfaces, which results in its irregular, unpredictable convergence (top row). Despite being based on MCMC sampling, our method (bottom row) shows good convergence properties typical of regular Monte Carlo algorithms: unconverged results provide a reliable preview of the final image and are amenable to denoising.

all path sampling techniques from VCM/UPS (which apart from bidirectional path tracing techniques also includes subpath reuse with density estimation, see Section 2.7).

Utilizing density estimation and subpath reuse in MCMC context. Fan et al. [2005] were the first to use MCMC to generate photons (vertices of light subpaths) to improve density estimation of photon-based methods (see Section 2.6) in scenes with complex visibility. While later several other works improved the distribution of photons generated by MCMC (see Sections 4.4.2 and 4.7.3), none of them utilizes all of VCM/UPS path sampling techniques (as our algorithm) and they cannot generate all possible light transport paths (e.g. pure specular paths).

Improving stratification of MCMC samples. Energy redistribution path tracing (see Section 4.8.1) strives to retain stratification by running many short Markov chains, where each is initialized by a path starting in a different image pixel. However, since the underlying algorithm is still equivalent to the original MLT (i.e. the used mutations and the target functions are the same), the algorithm is susceptible to chain being stuck at one sample for a long time, and thus the success has only been partial. We approach the issue of stratification differently, as discussed in Section 6.2).

Utilizing replica exchange. Replica exchange [Swendsen and Wang, 1986] is another important idea to reduce correlation in MCMC sampling that has been previously applied in light transport (see Section 4.5). In Chapter 5 we discuss in detail our systematic attempt at improving global exploration in MCMC algorithms in light transport. The algorithm described there uses replica exchange as the only key component and allows global exploration of hard-to-sample paths by tempering the BSDFs. Here we achieve better exploration of hard-to-sample

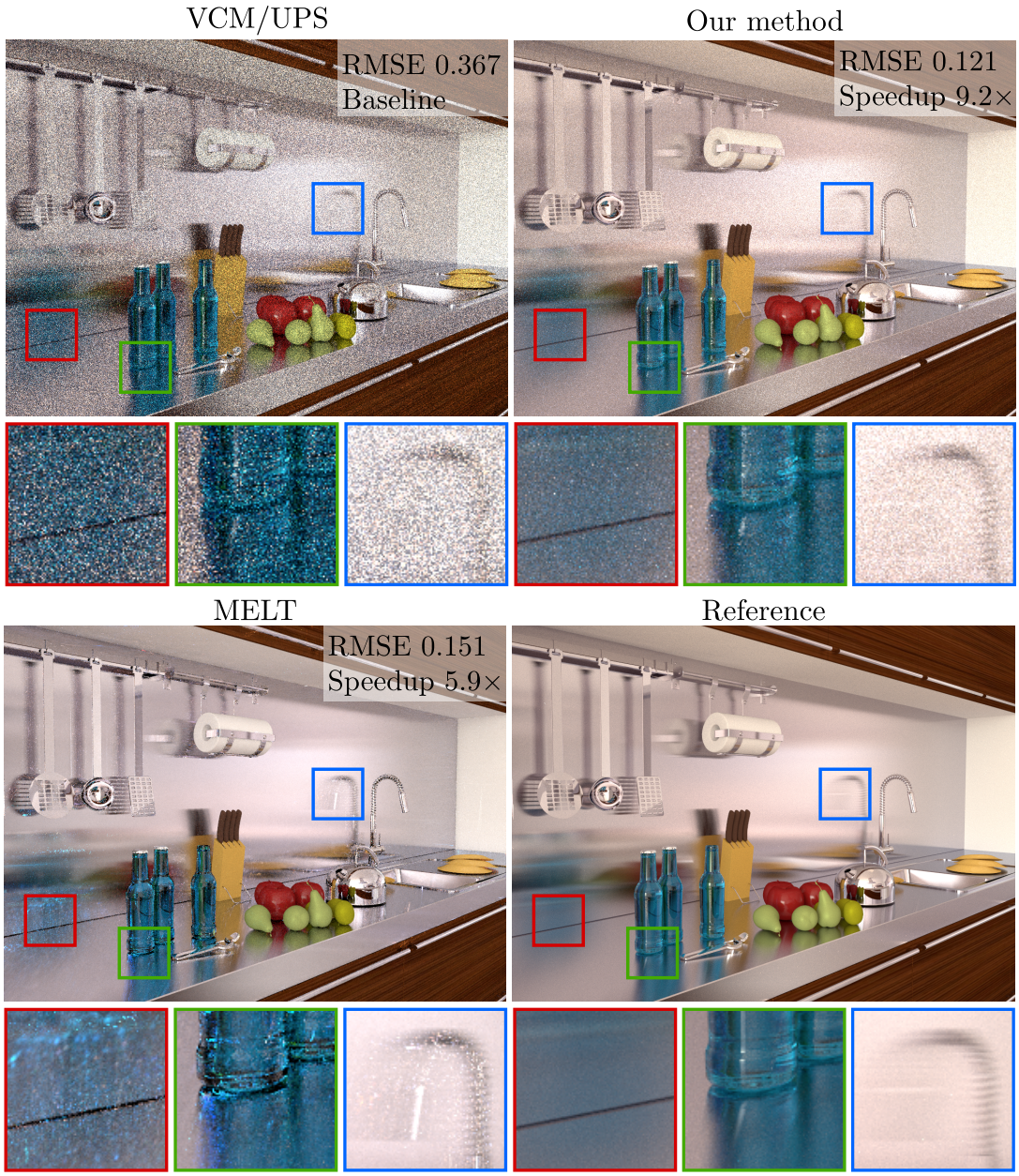

Figure 6.2: Our light transport simulation algorithm fuses key ideas behind VCM/UPS (see Section 2.7) and Markov chain Monte Carlo methods. It outperforms state-of-the-art approaches, such as VCM/UPS or manifold exploration light transport (MELT, see Section 4.3.1), especially in scenes with complex glossy or specular transport and complex visibility. At the same time, the algorithm is substantially simpler than many of the recent MCMC based light transport algorithms.

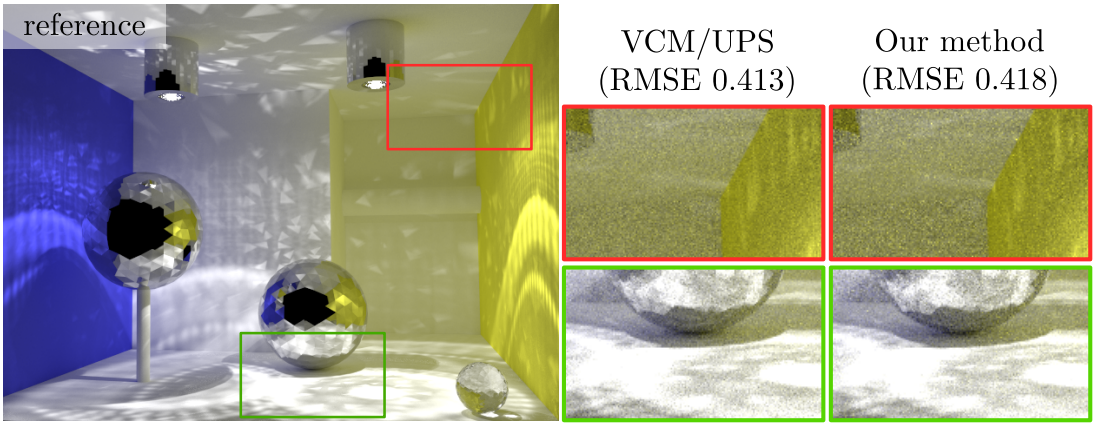

Figure 6.3: Our algorithm does not significantly impair the performance of VCM/UPS in simple scenes, where most of the light subpaths generated by independent Monte Carlo contribute to the image. We ran both methods for 5 minutes to render these images.

paths by utilizing all path sampling techniques from VCM/UPS, while only employing replica exchange to further improve the robustness of our algorithm.

6.2 Algorithm design

In this section, we summarize the design decisions that led to our algorithm and briefly mention possible alternatives.

6.2.1 Requirements

Our goal is to develop an MCMC algorithm that can efficiently render scenes with various kinds of (indirect) illumination, including *complex transport on glossy/specular surfaces*, while having a predictable convergence. At the same time, thanks to MCMC sampling the algorithm should maintain good efficiency in scenes with *complex visibility* between light sources and the camera, such as natural light illuminating an interior through a small glass window pane. For the algorithm to be practical, it must *not impair the efficiency* of the current state-of-the-art approaches *in the simple cases* where they already work well. And it should also be *relatively easy to implement* in existing rendering software to facilitate its adoption in practice.

6.2.2 General algorithm design strategy

We design the new algorithm based on ideas that have individually been shown to meet some of the above requirements. These include *bidirectional path estimators* and their *combination via multiple importance sampling (MIS)*, the vertex merging technique (i.e. photon density estimation) with its inherent *regularization* and large-scale *subpath reuse*. These ideas are the building blocks of VCM/UPS (see Section 2.7), which excels at handling complex glossy/specular transport, unlike simpler solutions such as path tracing. Furthermore, because VCM/UPS has been implemented in many popular rendering software solutions [Pixar, 1982]

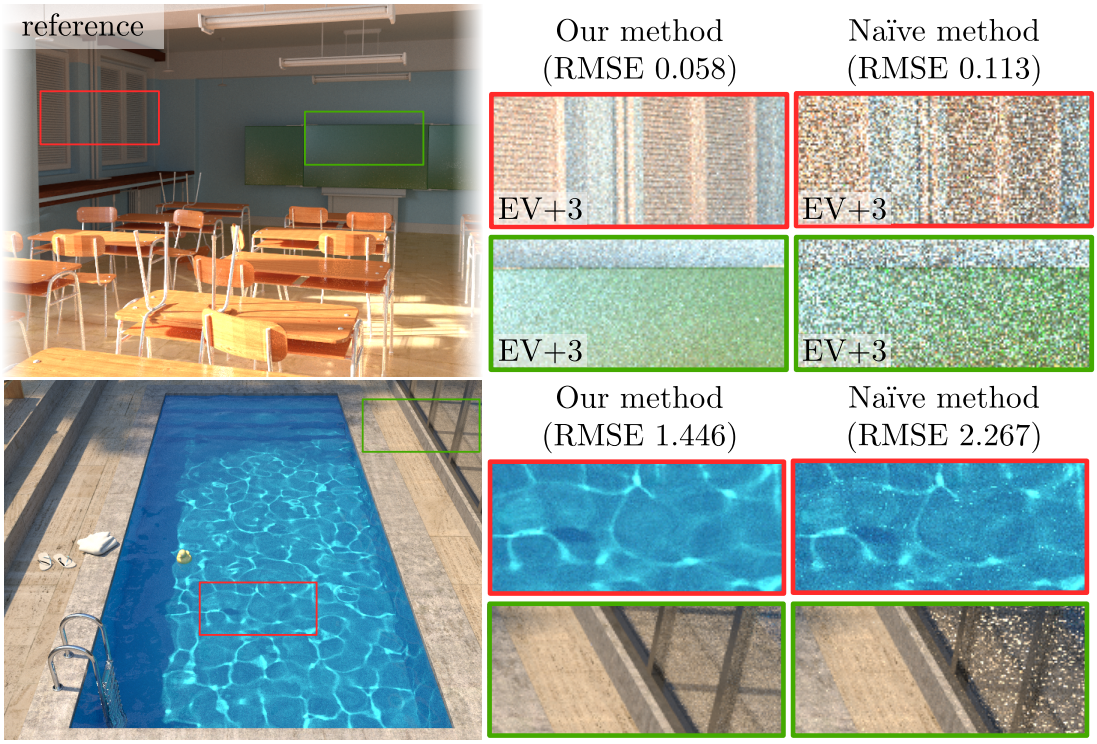

Figure 6.4: Equal-time comparison (1 hour) between our method and a naïve combination of PSSMLT (see Section 4.2.1) and VCM/UPS with photons generated by visibility-driven MCMC (see Section 4.4.2). EV gives the exposure value adjustment used for the insets.

and our final algorithm is a lightweight extension of it, our algorithm can be easily adopted in practice.

We pay attention not to inherit the shortcomings of VCM/UPS and MCMC sampling: bad performance of VCM/UPS in scenes with complex visibility due to light subpaths not being able to reach visually important regions; image artifacts and bad convergence behavior of MCMC due to poor global exploration; and insufficient image plane stratification in many existing MCMC approaches.

6.2.3 Design choices

We choose VCM/UPS as the basis of our solution while using a MCMC sampler for path sampling. This high-level idea, however, leaves a number of unanswered questions: **Q1**: For which paths (camera, light, or full paths) should the MCMC sampler be used? **Q2**: What specific MCMC sampler should be used and what state space should it operate in (path space, primary sample space, or other)? **Q3**: How should its target function be defined? The specific design choices we make are directly motivated by the requirements stated above.

Figure 6.4 illustrates the importance of finding suitable answers to these questions. We compare our method to a naïve algorithm that extends PSSMLT (see Section 4.2.1) by including all the sampling techniques from VCM/UPS. Light and camera subpaths are generated and combined as in original PSSMLT, but we use an additional set of light subpaths pre-generated by visibility-driven photon tracing of Hachisuka and Jensen (see Section 4.4.2) to enable the merging tech-

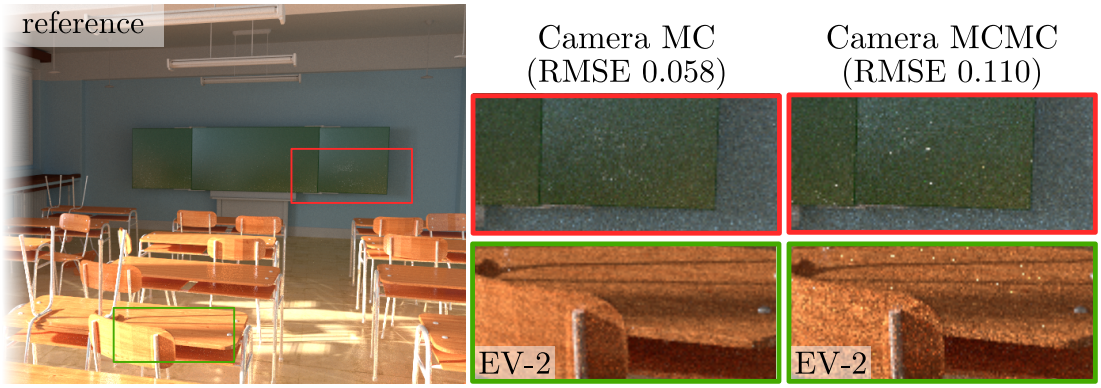

Figure 6.5: Comparison of camera subpath generation by stratified Monte Carlo sampling (left) and by a visibility-driven MCMC sampler (right) in our algorithm. Camera subpath generation via MCMC is outperformed by stratified sampling in this, as well as in other scenes. EV gives the exposure value adjustment used for the insets.

nique. The poor performance of this algorithm compared to our solution shows that even within the relatively narrowed-down scope of combining VCM/UPS and MCMC sampling, the design choices can make a big difference in the resulting algorithm's performance (Appendix 6.C discusses other possible combinations that we have considered in our research).

Q1: MCMC sampling of camera, light, or full paths. Previous work (see Sections 4.4.2, 4.6.1, and 4.7.3) has shown that using MCMC sampling to steer *light subpaths* towards visually important regions can have tremendous efficiency benefits; we follow this idea. Likewise, one could expect that applying the same idea to *camera subpaths* in a combined algorithm – which is essentially symmetric with respect to the camera and light sources – should yield a similar benefit. However, we found the contrary to be true, as shown in Figure 6.5. We explain this by the fact that high correlation of camera subpaths due to MCMC sampling compromises one of our design goals – good image plane stratification. For this reason, we opt for the use of regular, independent Monte Carlo to sample the camera subpaths. For the same reason, we also rule out the option of using MCMC sampling for *full* paths as in MLT. Figure 6.6 illustrates image plane stratification/sample correlation of different methods.

Q2: MCMC sampling space. Motivated by the goal of algorithmic simplicity, we choose as our base MCMC algorithm a Metropolis-Hastings sampler (enhanced via replica exchange, see Section 4.5)), that operates in the primary sample space (see Section 4.2.1). MCMC sampling directly in the path space would unnecessarily complicate the algorithm and go against one of our design principles. Furthermore, transformation from the primary sample space to the path space naturally importance-samples BSDFs and geometry terms along the path. Achieving this in MLT framework requires a rather complex transformation of paths from the path space to yet another space (see Section 4.3.2).

Q3: Target function. In designing the target function, we face two contradicting goals: on the one hand, a target function with a simple shape (such as path visibility) is easily explored by a MCMC sampler, which reduces sample correlation and its negative practical consequences. On the other hand, a target

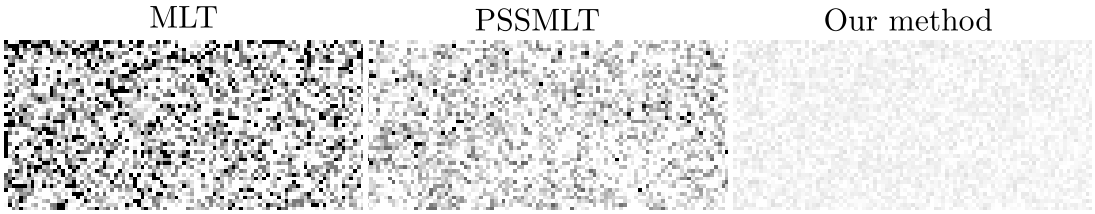

Figure 6.6: Equal-sample comparison (5 samples per pixel) of rendering uniformly lit rectangle. Ideally, the resulting image should have uniform color. The original MLT (see Section 4.1) delivers the worst results due to high sample correlation. While PSSMLT (see Section 4.2.1) produces a slightly better result than MLT, our algorithm shows improvement over both MLT and PSSMLT thanks to our choice of using stratified camera subpaths.

function more similar to the actual path contribution can potentially reduce variance as per the importance sampling principle; but it may also lead to excessive sample correlation. Given the complementary pros and cons of the two options, we use two Markov chains, one with *path contribution* and the other with *path visibility* as their target functions. We refer to the two chains as the *contribution chain* and the *visibility chain*, respectively. We let the two chains interact via replica exchange and we combine the samples produced by the two chains via MIS (see Section 2.3). This design contributes to our goal of reduced sample correlation in MCMC. Figure 6.7 illustrates the complementary advantages of the two target functions and the benefit of their combination.

6.3 Algorithm

This section provides a complete description of our algorithm, starting with an overview shown in Figure 6.8. The algorithm works in iterations, each of which starts by generating all camera subpaths by an independent Monte Carlo sampler (line 2). We store all their vertices so that the merging technique can later be evaluated during the MCMC light subpath sampling. The camera subpaths are used to immediately evaluate the contribution of unidirectional path tracing and path tracing with next event estimation, as these sampling techniques are independent of light subpaths[1] (line 3). We then execute our MCMC sampler, which is based on PSSMLT (see Section 4.2.1), to generate light subpaths and evaluate the remaining VCM/UPS techniques (line 4).

6.3.1 Light subpath contribution

Contribution definitions. Let us start by formally defining the *light subpath contribution* that is used to evaluate the target function for both the contribution chain and the visibility chain. A single light subpath $\overline{x}^{\mathcal{L}}$ in the VCM/UPS algorithm gives rise to an entire family of full paths created by light tracing, bid-

[1]Note that for path tracing with next event estimation we always generate a new position on a light source (for the deterministic connection) rather than using the first vertex of a light subpath. This makes the technique independent on the light subpath and in turn decreases the correlation.

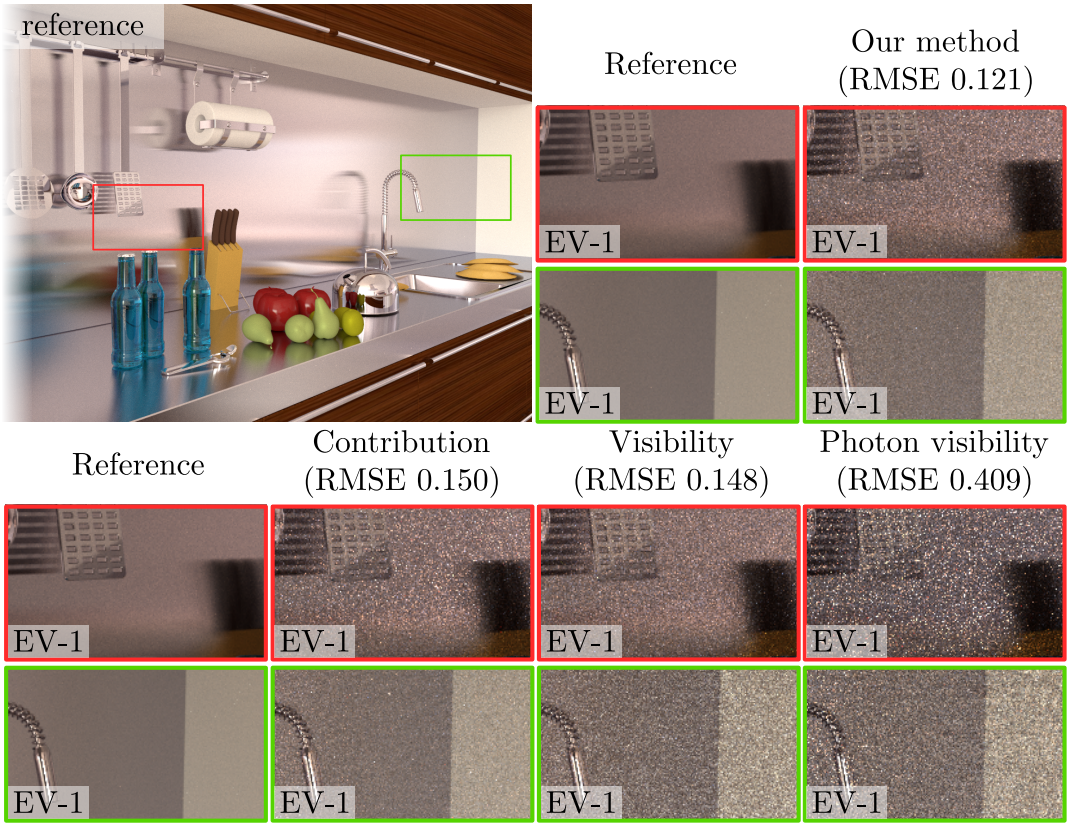

Figure 6.7: Comparison of different target functions in our algorithm. From right to left, a target function based solely on photon visibility (see Section 4.4.2) often yields suboptimal results. Our path visibility definition, which takes into account all the sampling techniques in VCM/UPS, considerably improves the results, but it may sometimes be outperformed by the target function given by the path contribution itself. Best overall results are achieved by our combined two-chain approach based on replica exchange.

irectional connections, and vertex merging (see Section 2.7). Therefore, we have multiple full paths generated by a single light subpath. We define the contribution of $\overline{x}^{\mathcal{L}}$ as the sum over the individual MIS-weighted contributions of all these full paths. Let $\mathcal{F}(\overline{x}^{\mathcal{L}})$ denote the set of such full paths with the corresponding path sampling techniques that created them. The contribution of $\overline{x}^{\mathcal{L}}$ is then defined as follows

$$C^{\Omega}(\overline{x}^{\mathcal{L}}) = \sum_{(\overline{x},m)\in\mathcal{F}(\overline{x}^{\mathcal{L}})} \frac{f(\overline{x})w_m(\overline{x})}{p_m(\overline{x})}. \tag{6.1}$$

1: **for** $i = 1 \dots M$ **do**
2: $\quad\mathcal{C}$:= mcCameraSubpathSampling(N)
3: $\quad$image += evaluatePathTracing($\mathcal{C}$)
4: $\quad$image += mcmcLightSubpathSampling($\mathcal{C}$)

Figure 6.8: High-level overview of our algorithm.

Here $\overline{x}$ is a full path, m is a VCM/UPS sampling technique with the pdf p_m, f is the standard path contribution function (see Section 2.1), and $w_m(\overline{x})$ is our adjusted VCM/UPS multiple importance sampling (MIS) weight described below (Section 6.3.4). $C^{\Omega}(\overline{x}^{\mathcal{L}})$ is designed to measure the *total* image contribution due to the light subpath $\overline{x}^{\mathcal{L}}$ and thus it does not contain the image pixel filter h_j. Note that since the contribution of unidirectional path tracing and path tracing with next event estimation is independent of $\overline{x}^{\mathcal{L}}$, the paths created by these techniques are not in $\mathcal{F}(\overline{x}^{\mathcal{L}})$.

Two target functions. We now use the light subpath contribution $C^{\Omega}(\overline{x}^{\mathcal{L}})$ to define the target function for the contribution and visibility Markov chains, respectively, as

$$\pi_{\text{con}}(C^{\Omega}(\overline{x}^{\mathcal{L}})) = \text{lum}(C^{\Omega}(\overline{x}^{\mathcal{L}})) \tag{6.2}$$

$$\pi_{\text{vis}}(C^{\Omega}(\overline{x}^{\mathcal{L}})) = \begin{cases} 1 & \text{if } C^{\Omega}(\overline{x}^{\mathcal{L}}) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

That is, the former is given by the scalar luminance of the subpath contribution, while the latter is a binary indicator equal to one for light subpaths that make a non-zero contribution to the image.

Computing contribution. Contribution of a light subpath $C^{\Omega}(\overline{x}^{\mathcal{L}})$ consists of the contributions by bidirectional connections and by vertex merging. The merging contribution is computed using all pre-generated camera subpaths $\mathcal{C}$. In order to compute bidirectional connections we pick (at random) one camera subpath from $\mathcal{C}$ per each light subpath. As we already mentioned, we exclude the contribution by path tracing techniques from $C^{\Omega}(\overline{x}^{\mathcal{L}})$. Figure 6.9 shows how we combine the contributions from light and camera subpaths.

6.3.2 MCMC sampler details

We now provide details of our MCMC sampler used to generate light subpaths.

Initialization. To initialize our Markov chains, we apply the start-up bias elimination method (see Section 3.6). The method draws an initial state using importance resampling from a small set (10000) of independent light subpaths.

Mutations. We use primary sample space MCMC, where we mutate a primary space sample $\overline{u} \in \mathcal{P}$ using an adaptive mutation kernel (see Appendix 6.B for more details). Using $\overline{u}$ as a vector of random numbers we then generate a light subpath $\overline{x}^{\mathcal{L}}$ via local sampling techniques (i.e. light source emission and BSDF importance sampling). As in PSSMLT (see Section 4.2.1), we alternate between small steps (local mutations) and large steps (independent global mutations) to ensure both good local and global exploration. However, the large steps are used only in the visibility chain, because replica exchange between the chains is sufficient to ensure global exploration in the contribution chain. In all scenes, we use the large step with probability $p(\text{largeStep}) = 0.3$.

Normalization. To estimate light transport using an MCMC sampler, we need the normalization constant $b_c = \int_{\mathcal{P}} \pi_c(C^{\mathcal{P}}(\overline{u}))\mathrm{d}\overline{u}$ for both chains. Here, we have introduced the notation $C^{\mathcal{P}}(\overline{u}) = C^{\Omega}(\overline{x}^{\mathcal{L}})$, where $\overline{x}^{\mathcal{L}}$ is the light subpath generated from the primary sample $\overline{u}$.

Since we use start-up bias elimination method (see Section 3.6), we should use the weight b_c^* computed from the initial sample set instead of normalization.

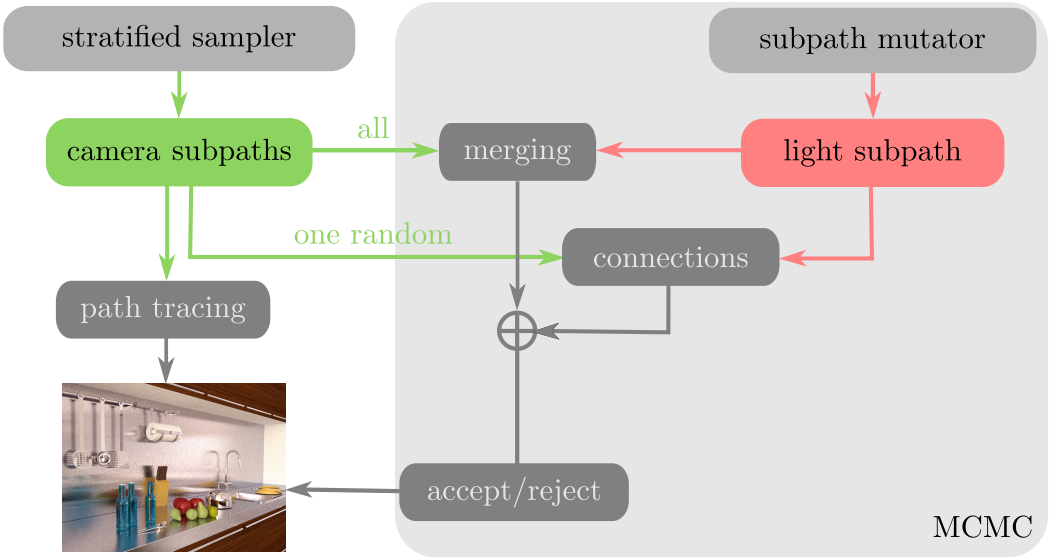

Figure 6.9: Diagram showing how the contributions of light and camera subpaths are combined to compute the image. Each light subpath is first splatted onto all camera subpath vertices (merging) and then connected to a randomly selected camera subpath. The contributions due to both merging and connections are then combined via MIS weights (see Section 2.3) and used for accepting/rejecting the light subpath as well as for computing the image. While the contribution by path tracing techniques is computed outside of the MCMC sampler, it also undergoes MIS weighting.

However, we have find out that using this weight in practice may lead to darker images if the scene has complex visibility. Thus we compute the normalization using a regular Monte Carlo estimator, which utilizes the large steps of our MCMC algorithm. Given L large steps, the normalization is estimated as follows

$$\langle b_c \rangle = \frac{1}{L} \sum_{j=1}^{L} \pi_c(C^\Omega(\overline{x}_j^\mathcal{L})). \tag{6.4}$$

We keep updating this estimate during the entire rendering process.

Combining the two chains. Using only samples from the contribution chain to render the image would be wasteful as half of the samples would be thrown away. Section 5.4 discusses that we can apply another set of MIS weights (different from those used for VCM/UPS in Equation (6.1)) in order to combine the contributions from all chains. Plugging our visibility and contribution chains into Equation (5.13) we receive a following MCMC MIS weight for a given light subpath $\overline{x}^\mathcal{L}$ from a chain c

$$w_{\text{mcmc}}(c, \overline{x}^\mathcal{L}) = \frac{\frac{\pi_c(C^\Omega)}{b_c}}{\frac{\pi_{\text{vis}}(C^\Omega)}{b_{\text{vis}}} + \frac{\pi_{\text{con}}(C^\Omega)}{b_{\text{con}}}}, \tag{6.5}$$

where we have dropped the argument of $C^\Omega(\overline{x}^\mathcal{L})$ for the sake of clarity. b_c is the normalization constant of the chain $c \in \{\text{con}, \text{vis}\}$. To compute a consistent MIS weights throughout a given iteration, we use the normalization constants from the beginning of the iteration.

Restarting MCMC chains. Since the contribution $C^{\Omega}(\overline{x}^{\mathcal{L}})$ depends on camera subpaths $\mathcal{C}$, both target functions change when we generate a new set $\mathcal{C}$. To ensure convergence, we restart the MCMC chain at the beginning of each iteration. We then average the results of individual iterations to get the final result. The same algorithm was employed by Hachisuka and Jensen for photon tracing (see Section 4.4.2).

6.3.3 MCMC sampler implementation

Figure 6.10 provides a detailed pseudocode of our MCMC algorithm used to sample light subpaths. It receives the precomputed set of camera subpaths $\mathcal{C}$, and starts by initializing the two Markov chains (line 1). The loop generating individual light subpaths starts by picking (at random) the camera subpath $\overline{x}^{\mathcal{C}}$ (line 3) that the next light subpath will connect to. After that, we select the Markov chain c to update (line 4): we alternate between the contribution (con) and visibility (vis) chains, both receiving the same number of samples.

To propose a candidate $\overline{u}$ for the selected chain c, we mutate the current state $\overline{u}_{c,j-1}$ in the primary sample space (line 5). Once we have the primary sample space candidate $\overline{u}$, we map it to a light subpath $\overline{x}^{\mathcal{L}}$ by using a standard photon tracing algorithm that uses $\overline{u}$ as a vector of random numbers (line 6). The light subpath contribution C^{Ω} is then calculated using Equation (6.1) (line 7). We then use the contribution C^{Ω} to evaluate the target function for the proposed light subpath using Equation (6.2) or (6.3), which is, in turn, used to calculate the acceptance probability α (line 8). Based on that probability, we either accept or reject the proposed light subpath (lines 10–12).

We accumulate the contributions of both the proposed and the current light subpaths (lines 13–14), as per the use of expected values (see Section 3.7). To combine the contributions from the contribution and visibility chains, we apply the MCMC MIS weights defined by Equation (6.5). The replica exchange probability is calculated on line 15 by Equation (4.11), simplified by the fact that $\pi_{\text{vis}}(C^{\Omega}(\overline{x}^{\mathcal{L}})) = 1$ for any accepted light subpath. The probabilistic state exchange is then carried out on line 17. Finally, whenever an independent large step is performed, we update normalization constant estimates $\langle b_{\text{vis}} \rangle$ and $\langle b_{\text{con}} \rangle$ as described by Equation (6.4) (line 19). While large steps are only generated for the visibility chain, we can use them to progressively update normalization constants of both chains during the entire rendering process.

After having generated N light subpaths in the current iteration, we combine their contributions and output the progress of rendering (line 20), while dividing by the number of iterations M.

6.3.4 MIS weight computation

We now return to the computation of MIS weights for combining the VCM/UPS path sampling techniques in Equation (6.1). A standard approach is to use the balance heuristic Equation 2.6.

To compute balance heuristic we must know the effective pdf with which the full path $\overline{x}$ would be sampled by the path sampling technique m. Unlike in standard Monte Carlo, in our algorithm the pdf depends on the MCMC sampler.

$[\overline{u}_{\text{con},0}, \overline{u}_{\text{vis},0}] :=$ INITIALIZECHAINS()
for $j = 1 \dots N$ **do**
  $\overline{x}^{\mathcal{C}} :=$ GETNEXTCAMERAPATH($\mathcal{C}$)
  Select $c \in \{\text{con}, \text{vis}\}$ ▷ alternate between contrib. and visibility chains
  $\overline{u} :=$ MUTATE($\overline{u}_{c,j-1}$) ▷ propose candidate via mutation
  $\overline{x}^{\mathcal{L}} :=$ TRACELIGHTSUBPATH($\overline{u}$)
  $C^{\Omega} :=$ COMPUTESUBPATHCONTRIB($\overline{x}^{\mathcal{L}}$, $\overline{x}^{\mathcal{C}}$, $\mathcal{C}$) ▷ Equation (6.1)
  $\alpha :=$ CALCACCEPTANCEPROB($\pi_c(C^{\Omega})$, $\pi_c(C^{\Omega}_{c,j-1})$) ▷ Equation (3.2)
  if $\alpha >$ RANDOM() **then**
    $[\overline{u}_{c,j}, C^{\Omega}_{c,j}] := [\overline{u}, C^{\Omega}]$ ▷ accept mutation
  else
    $[\overline{u}_{c,j}, C^{\Omega}_{c,j}] := [\overline{u}_{c,j-1}, C^{\Omega}_{c,j-1}]$ ▷ reject mutation, keep old state
  $\text{image}_c \mathrel{+}= (1-\alpha) \times \frac{C^{\Omega}_{c,j-1}}{\pi_c(C^{\Omega}_{c,j-1})} \times$ MCMCMISWEIGHT($C^{\Omega}_{c,j-1}$)
  $\text{image}_c \mathrel{+}= \alpha \times \frac{C^{\Omega}}{\pi_c(C^{\Omega})} \times$ MCMCMISWEIGHT(C^{Ω})
  $r(\text{con}, \text{vis}) := \min\left\{1, \frac{\pi_{\text{con}}(C^{\Omega}_{\text{vis},j})}{\pi_{\text{con}}(C^{\Omega}_{\text{con},j})}\right\}$ ▷ exchange probability, Equation (4.11)
  if $r(\text{con}, \text{vis}) >$ RANDOM() **then**
    SWAP($[\overline{u}_{\text{vis},j}, C^{\Omega}_{\text{vis},j}]$, $[\overline{u}_{\text{con},j}, C^{\Omega}_{\text{con},j}]$) ▷ replica exchange
  if largeStep = true **then**
    UPDATENORMALIZATION(C^{Ω}) ▷ Equation (6.4)
OUTPUT($2\frac{(b_{\text{vis}}\text{image}_{\text{vis}} + b_{\text{con}}\text{image}_{\text{con}})}{NM}$) ▷ progressive output

Figure 6.10: Our MCMC algorithm for generating light subpaths.

This makes the computation of the pdf in our algorithm unfeasible in practice. This is a shared limitation with PSSMLT (see Section 4.2.1), and has not been reported by previous work. The issue lies in the fact that given a fixed path sampling technique m a full path $\overline{x}$ can be generated by infinitely many different primary samples $\mathcal{P}_{\overline{x}} \subseteq \mathcal{P}$, each of which corresponds to a *light* subpath with the same prefix (see Figure 6.11). Furthermore, each such $\overline{u} \in \mathcal{P}_{\overline{x}}$ usually has a different target function value, since it is given by the sum of the contributions of all full paths generated by the light subpath. The exact pdf $\hat{p}_m(\overline{x})$ is therefore given by

$$\hat{p}_m(\overline{x}) = p_m(\overline{x}) p_c^{\mathcal{P}}(\overline{x}) = p_m(\overline{x}) \int_{\mathcal{P}_{\overline{x}}} \frac{\pi_c(C^{\mathcal{P}}(\overline{u}))}{b_c} \mathrm{d}\mu_{\overline{x}}(\overline{u}). \tag{6.6}$$

Here $p_m(\overline{x})$ is the pdf of the path sampling technique m, i.e. the inverse of the Jacobian of mapping $\overline{u}$ to $\overline{x}$, which is the same for all $\overline{u} \in \mathcal{P}_{\overline{x}}$. $p_c^{\mathcal{P}}(\overline{x})$ is the marginalized probability density of sampling $\overline{u} \in \mathcal{P}_{\overline{x}}$ by the Markov chain c. $\pi_c(C^{\mathcal{P}}(\overline{u}))$ is the target function of the chain c, and b_c is the normalization of the chain c. The measure function $\mu_{\overline{x}}(\overline{u})$ is then a simple volume measure which ignores the common prefix of all $\overline{u} \in \mathcal{P}_{\overline{x}}$.

PSSMLT approximates the pdf in Equation (6.6) by ignoring $p_c^{\mathcal{P}}(\overline{x})$ (i.e. $\hat{p}_m(\overline{x}) \approx p_m(\overline{x})$). While such an approach is suboptimal, because the MIS weights do not use the actual pdf of the paths, it usually works well in practice. For this reason, we adopt it for paths generated by our *contribution chain* as well.

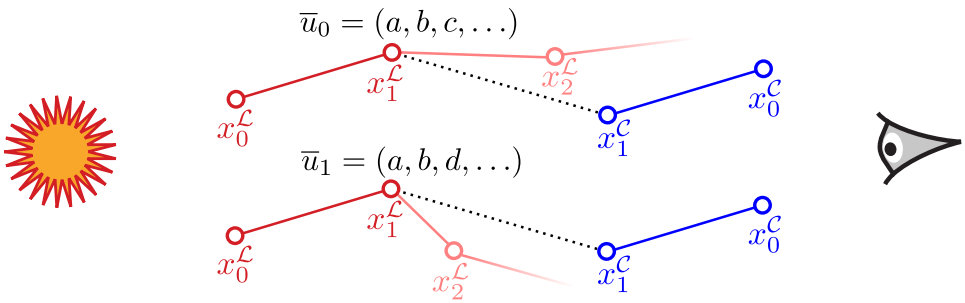

Figure 6.11: A single full path $\overline{x} = (x_0^{\mathcal{L}}, x_1^{\mathcal{L}}, x_1^{\mathcal{C}}, x_0^{\mathcal{C}})$ can be generated by infinitely many different primary samples ($\overline{u}_0$ and $\overline{u}_1$ in this example) that all share the same prefix (a, b). The remaining components of each primary sample ($c, \ldots$ and $d, \ldots$ in this example) are used to generate the vertices of the light subpath after $x_1^{\mathcal{L}}$. Those are *not* part of the full path, but still affect its pdf, since the MCMC sampler accepts/rejects a whole batch of full paths that share the same primary sample.

However, we show that for the *visibility chain*, we can compute the exact path pdf and thus improve the algorithm efficiency (see Figure 6.12). While a full path $\overline{x}$ can be constructed from infinitely many primary samples $\mathcal{P}_{\overline{x}}$ of the visibility chain, they all have the same target function value equal to one [2]. Given the above and the fact that a subspace of $\mathcal{P}_{\overline{x}}$ defined by $\mu_{\overline{x}}(\overline{u})$ has a unit volume, the exact pdf $\hat{p}_m(\overline{x})$ is computed as

$$\hat{p}_m(\overline{x}) = p_m(\overline{x}) \int_{\mathcal{P}_{\overline{x}}} \frac{\mathrm{d}\mu_{\overline{x}}(\overline{u})}{b_{\text{vis}}} = \frac{p_m(\overline{x})}{b_{\text{vis}}}. \tag{6.7}$$

Using the exact pdfs compared to the approximated pdfs therefore results in scaling up the MIS weights of all the path sampling techniques that depend on light subpaths. The paths generated by the path tracing techniques are independent of the MCMC process, and thus their pdfs are computed as in bidirectional path tracing, $\hat{p}_m(\overline{x}) = p_m(\overline{x})$.

6.4 Results

We have implemented our algorithm and baseline VCM/UPS on top of the Mitsuba renderer [Jakob, 2010]. The code is available on the project web page (see Attachments). We ran all the experiments on a PC with an Intel Core i7 at 3.50 GHz with 32 GB RAM using eight logical cores. All the reference images were rendered using the VCM/UPS algorithm over the course of two weeks.

Among the previous MCMC algorithms we have tested, manifold exploration light transport (MELT, see Section 4.3.1) provided the best results which is why we use it for most of our comparisons in this chapter. Note that additional results not included here can be found on the project web page (see Attachments).

MELT parameters used in Mitsuba have been manually tuned for each scene to ensure the best performance. We used adaptive mutation size (see Appendix 6.B)

[2] We compute MIS weight only for a path $\overline{x}$ that has non-zero contribution and thus all primary samples in $\mathcal{P}_{\overline{x}}$ are mapped to a path with non-zero contribution and in the visibility chain they have target function value equal to one.

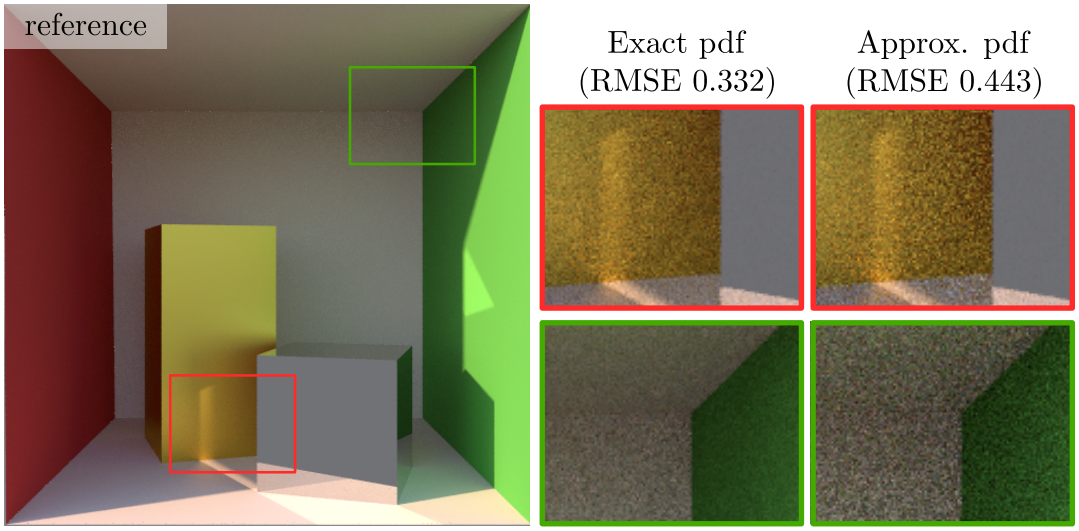

Figure 6.12: Equal-sample (64 iterations) comparison of results generated using only the visibility chain with the exact pdfs used for MIS weights computation (left) and with the approximated pdfs (right). Using exact pdfs for MIS weights computation decreases noise in the image due to better weighting of sampling techniques.

for our algorithm, PSSMLT (see Section 4.2.1), and multiplexed MLT (MMLT, see Section 4.2.2) with the target acceptance rate of 23.4% (this is considered optimal under certain conditions [Rosenthal, 2011]), which is roughly attained in our tests.

Radii for the vertex merging (i.e. photon density estimation) techniques at all the vertices of a camera subpath are determined by the pixel footprint at the first vertex of the respective subpath. We have not used any radius reduction [Hachisuka et al., 2008] but we have experimentally verified that the radius is small enough so that any bias is numerically and visually negligible in our scenes. Each iteration of our algorithm and baseline VCM/UPS uses one camera and light subpath per pixel.

Comparison to VCM/UPS and MLT. Figures 6.2,6.13, 6.14 show equal-time (one hour) comparisons of our algorithm, baseline VCM/UPS, and MELT. All three cases represent a typical scene configuration where an interior is illuminated from the outside via an opening such as a window. Table 6.1 summarizes rendering parameters and statistics.

Overall, the performance of baseline VCM/UPS suffers from an insufficient number of light subpaths due to complex visibility. Even for the **Living room** scene which features a relatively large window and is thus easier to render with

	Kitchen	**Living room**	**Classroom**
Image resolution	1024×768	1024×768	960×480
#iterations VCM/UPS	675	641	1002
#iterations our algorithm	357	406	525
avg. exchange rate (our algo.)	26.5%	26.4%	27.6%

Table 6.1: Settings and statistics for scenes in Figures 6.2,6.13, 6.14.

Figure 6.13: Equal-time comparisons (1 hour) of our method in the living room scene, baseline VCM/UPS (see Section 2.7), and manifold exploration light transport (MELT, see Section 4.3.1). The reported speedup is calculated as the squared ratio of RMSE values and gives the expected relative time that the baseline VCM/UPS would take to attain the RMSE of each respective algorithm. EV is the exposure value adjustment of the insets over the base image. The main text discusses the important features of the results shown here.

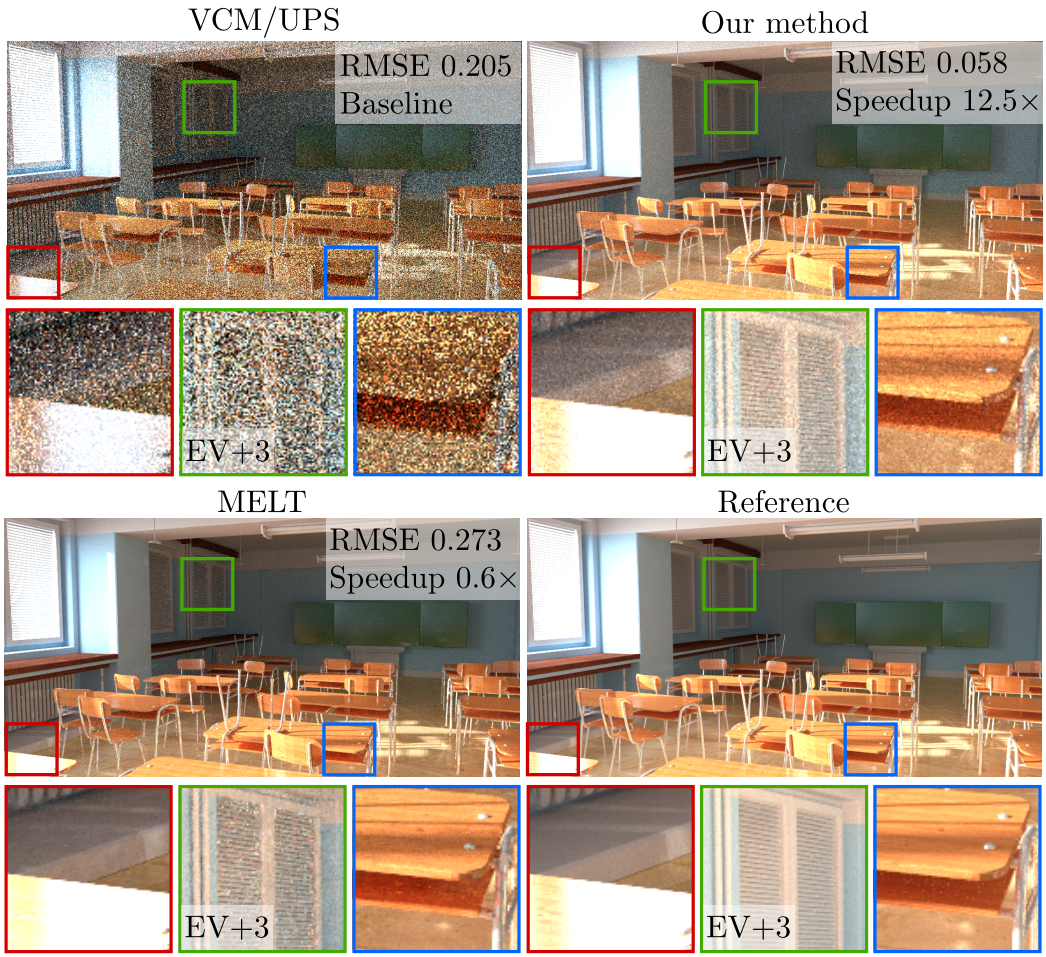

Figure 6.14: Equal-time comparisons (1 hour) of our method in the classroom scene, baseline VCM/UPS (see Section 2.7), and manifold exploration light transport (MELT, see Section 4.3.1). The reported speedup is calculated as the squared ratio of RMSE values and gives the expected relative time that the baseline VCM/UPS would take to attain the RMSE of each respective algorithm. EV is the exposure value adjustment of the insets over the base image. The main text discusses the important features of the results shown here.

baseline VCM/UPS, our algorithm achieves a speedup of more than 4×. MELT tends to fail to evenly explore the path space due to the presence of highly glossy/specular reflections. For example, the insets in Figure 6.13 (the **Living room** scene) highlight how MELT can either completely miss or over-explore specular-diffuse-specular paths. Our algorithm consistently outperforms both of the tested state-of-the-art algorithms in these comparisons.

The **Kitchen** scene in Figure 6.2 features a counter-top of highly anisotropic metal which often leads to various image artifacts due to sample correlation in MELT. Figure 6.15 shows RMSE plots within the insets of this scene, highlighting MELT's uneven convergence. Our algorithm steadily converges toward the correct solution within all the insets.

While the images rendered by MELT tend to be less noisy than from other algorithms, they are not necessarily converged to the correct solution. Figure 6.16 highlights this problem in the **Living room** and **Clasroom** scenes. While the noise-free image of MELT in Figures 6.13 and 6.14 could be mistakenly considered converged, its RMS error as well as the visualization in Figure 6.16 show that the contrary is true.

Comparison to other state-of-the-art algorithms. Figure 6.17 shows a comparison of our method to other state-of-the-art MCMC-based algorithms in the Kitchen scene (similar results for other scenes can be found on the project web page, see Attachments). Namely, we compare our method to primary sample

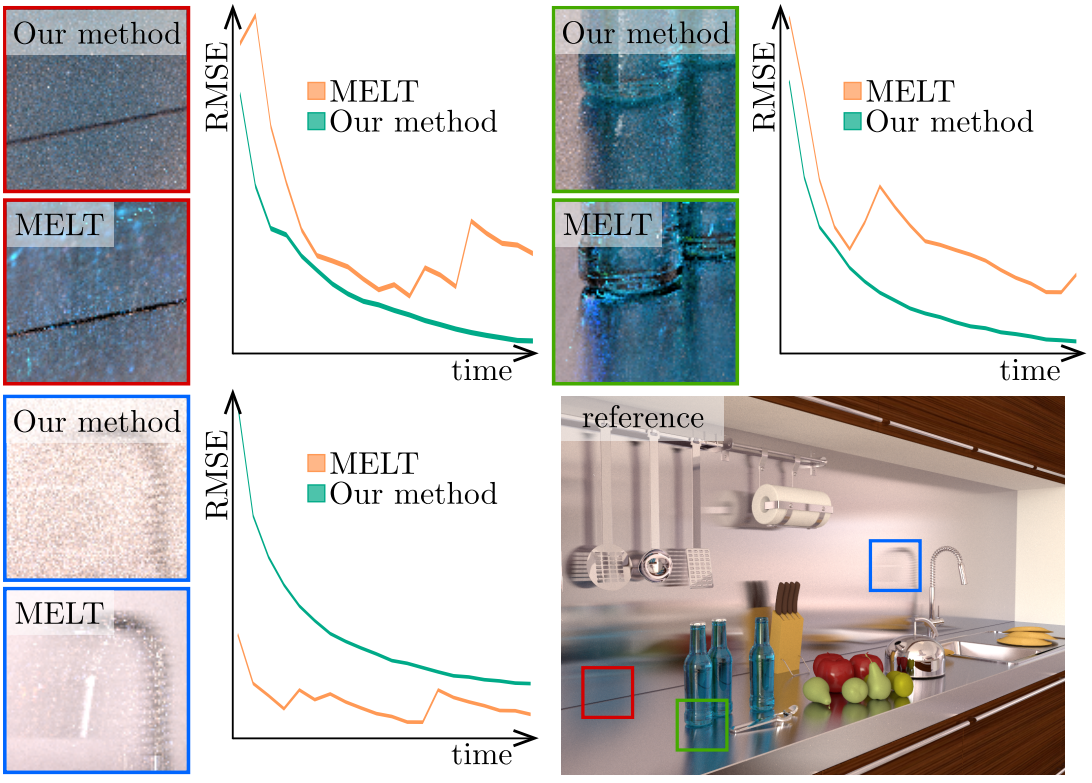

Figure 6.15: RMS error plots of our method and manifold exploration light transport (MELT, see Section 4.3.1) calculated for the insets of the Kitchen scene in Figure 6.2. While MELT may have lower numerical error than our method in some cases, its convergence behavior can be highly unpredictable. Our method predictably converges toward a noise-free solution.

Figure 6.16: Positive (red)-negative (green) difference of manifold exploration light transport (MELT, see Section 4.3.1) results in Figures 6.13 and 6.14 from the respective references for the Living room (left) and Classroom (right) scenes. Note that the difference image between correct result and reference should contain only uniform red-green noise, not the large red or green patches visible in these images.

space MLT (PSSMLT, see Section 4.2.1), multiplexed MLT (MMLT, see Section 4.2.2), manifold exploration light transport (MELT, see Section 4.3.1) with additional regularization (see Section 4.5.4), energy redistribution path tracing with manifold exploration (ERPT + ME, see Section 4.8.1), and our algorithm from Chapter 5 built on top of PSSMLT. For ERPT + ME we set the average number of chains per pixel to one and limit the number of mutations per chain to 100 (default setting in Mitsuba). ERPT + ME strives to improve stratification compared to MELT, but its short Markov chains impair its efficiency. For the regularized MELT, we use our own implementation, which performs regularization of interactions on highly glossy materials by adjusting their roughness (see Section 5.2). The regularized MELT allows easier sampling of some types of paths, however it does not improve MELT's irregular convergence. In fact, the results show that all the state-of-the-art algorithms suffer from severe image artifacts due to poor global exploration and resulting high sample correlation. While our algorithm from Chapter 5 seems to have better global exploration, in this case it fails to deliver a completely artifact-free image.

Figure 6.18 shows a comparison of our method to half vector space light transport (HSLT, see Section 4.3.2) in a variant of the Kitchen scene. Since the publicly available implementation of HSLT (in Mitsuba) cannot handle anisotropic materials, we set all materials to be isotropic for the sake of this specific comparison. We have set the universal perturbation probability in HSLT to get an overall 23.4% acceptance ratio. While previous work has reported better results for HSLT than MELT in some scenes (see Figure 4.13), we have not been able to observe any such improvement in our test scenes (even though we have consulted HSLT settings with the authors). In this specific example, we see that HSLT shares the issues of other MCMC methods: it suffers from irregular convergence and fails to globally explore glossy/specular light transport.

Consistency and behavior in simple scenes. Figure 6.19 plots the RMSE of our algorithm over 20 hours of rendering in the Kitchen scene. The RMSE steadily approaches to zero, which provides an empirical evidence of the convergence of our method.

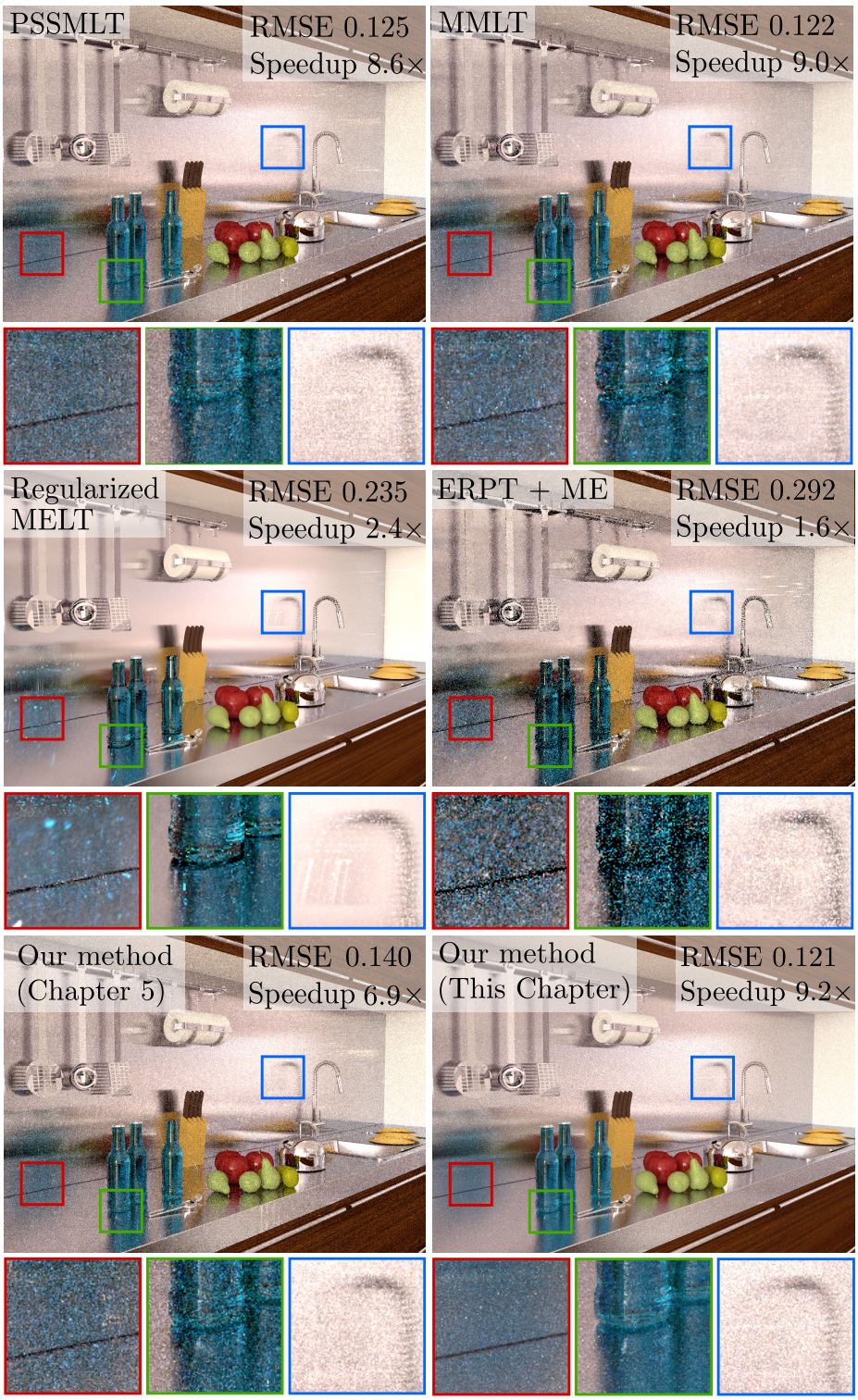

Figure 6.17: An equal-time (one hour) comparison of state-of-the-art MCMC-based algorithms in the Kitchen scene. The algorithms shown are primary sample space MLT (PSSMLT, see Section 4.2.1), multiplexed MLT (MMLT, see Section 4.2.2), manifold exploration light transport (MELT, see Section 4.3.1) with additional regularization (see Section 4.5.4), energy redistribution path tracing with manifold exploration (ERPT + ME, see Section 4.8.1), and our algorithm from Chapter 5 built on top of PSSMLT. The speedup is calculated as the squared ratio of RMSE values, and gives the expected relative time that the baseline VCM/UPS would take to attain the RMSE of each respective algorithm. Refer to Figure 6.2 for the results of baseline VCM/UPS.

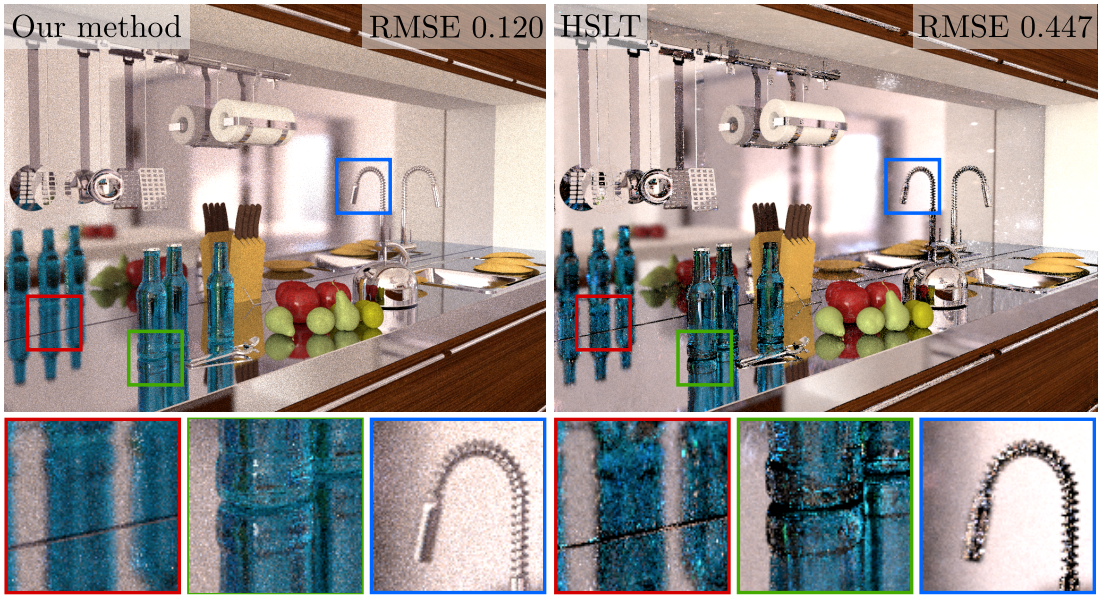

Figure 6.18: An equal-time (one hour) comparison of our method and half vector space light transport (HSLT, see Section 4.3.2) in the Kitchen scene with isotropic materials. We use this comparison because the HSLT implementation in Mitsuba cannot handle anisotropic materials. Reference can be found on the project web page (see Attachments).

We have already mentioned the low overhead of our algorithm over baseline VCM/UPS in simple scenes as presented in Figure 6.3. Figure 6.20 additionally demonstrates this in a comparison with primary sample space MLT (PSSMLT, see Section 4.2.1). Simpler algorithms like path tracing would outperform both our algorithm and VCM/UPS in a very simple scenes, however it is at the cost of their lower robustness.

Temporal coherence. To compare temporal coherence of PSSMLT, MELT, VCM/UPS and our algorithm, we have generated videos (using equal time renderings) showing a short flythrough of the Kitchen scene (the videos can be found on the project web page (see Attachments)). While PSSMLT and MELT exhibit pronounced flickering artifacts (i.e. low frequency noise), our results are substantially more well-behaved, only showing the high-frequency noise typical for regular Monte Carlo rendering. The results of VCM/UPS maintain good temporal coherence, however the overall noise level is very high.

Convergence analysis of our algorithm. In order to analyze which parts of our algorithm are responsible for its robustness and uniform convergence, we have run our algorithm with three different features switched on or off. Figure 6.21 describes the different versions and shows their RMSE plots (the images rendered by the different versions are then shown in Appendix 6.A). We can observe that some of the plots are not monotonic, which is caused by the algorithm getting stuck at local maxima and thus temporarily diverging from the reference. Clearly, the versions of our algorithm that combine two chains via replica exchange (green lines) are more robust to this problem. The plots also show that while using stratified camera subpaths lowers RMSE compared to using camera subpaths generated by MCMC, it does not guarantee monotonicity of the convergence plot. Using all VCM/UPS techniques instead of just bidirectional path tracing

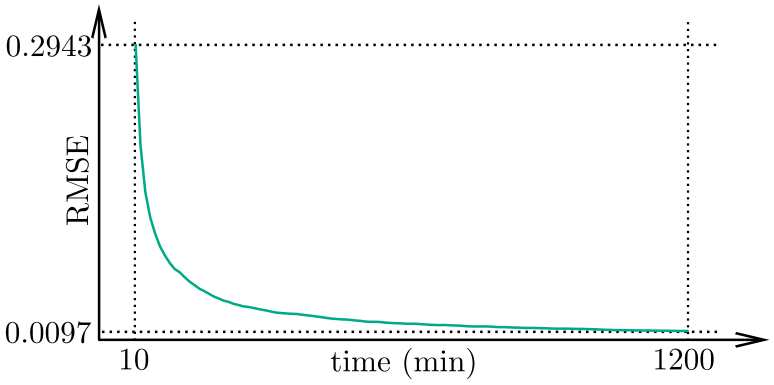

Figure 6.19: Empirical verification of our algorithm's convergence in the Kitchen scene. The figure shows a steadily decreasing plot of RMSE during 20 hours of rendering.

techniques then in most cases results in lower RMSE. We therefore conclude that the uniform convergence of our algorithm is mainly due to the use of replica exchange.

6.5 Limitations and future work

Though our algorithm robustly handles various scene configurations that are inefficient to render with existing algorithms, there are several limitations that need to be resolved by further work.

Negative effects of independent camera subpaths. While using stratified, independently generated camera subpaths is one of the key ideas behind the efficiency of our algorithm, it can negatively affect local exploration in our MCMC sampler. Consider a light subpath that makes a contribution only via a single bidirectional connection to the currently selected camera subpath. Even when

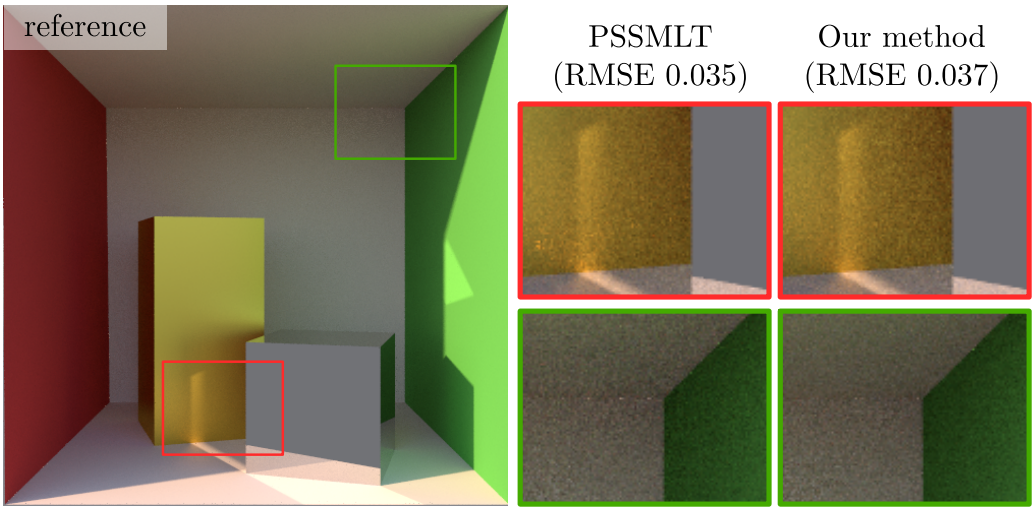

Figure 6.20: Our method has negligible overhead over primary sample space MLT (PSSMLT, see Section 4.2.1) in this simple scene, where the latter algorithm works well. The images took 2 minutes to render.

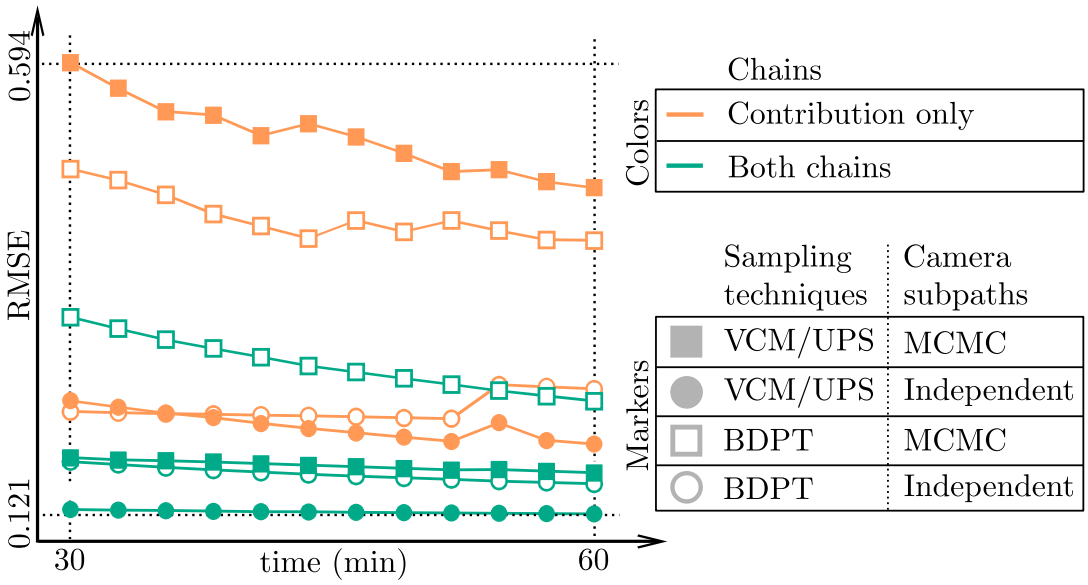

Figure 6.21: Here we show convergence plots for our algorithm in the Kitchen scene with three different features switched on or off. First, we can run our algorithm only with the contribution chain (orange lines) or with both chains using replica exchange (green lines). Second, we can either enable all VCM/UPS techniques (full markers) or enable only bidirectional path tracing (BDPT) techniques (empty markers). Finally, we can generate camera subpaths using a MCMC algorithm (square markers) or independent Monte Carlo (circle markers). We can observe from the plots that all components of our algorithm (namely replica exchange, all VCM/UPS techniques, independent camera subpaths) contribute to achieving the low RMSE. The plots are further discussed in the text.

the MCMC sampler proposes only a slight mutation of the *light* subpath, our algorithm still takes a completely independent *camera* subpath to connect to. As such, the previously successful bidirectional connection is likely to become invalid. Fortunately, this negative effect is diminished by the presence of vertex merging techniques, which do not depend on the currently selected camera path at all (see Figure 6.22).

Sub-optimal MIS weights. Our MIS weight calculation takes an approximated pdf (as in PSSMLT) for paths generated by the contribution chain (see Section 6.3.4). Though the practical impact of this approximation is unknown, since it is unfeasible to compute the exact pdf, deriving new MIS weights based on true pdfs with which our algorithm samples the paths would certainly increase its efficiency. Furthermore, we currently base our MIS weights on balance heuristic, which has been proven to work well under the condition of independent samples. This condition is however broken for MCMC and therefore using balance heuristic might be sub-optimal.

Specular paths. One limitation of our algorithm, shared by other approaches, is handling specular paths. That is, paths that interact *only* with highly specular

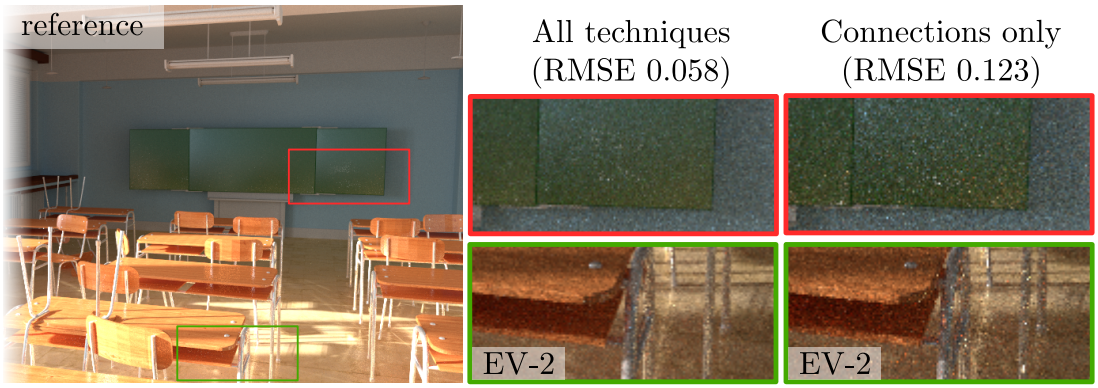

Figure 6.22: Equal-samples comparison (525 iterations) of our method using all VCM/UPS techniques and using only bidirectional techniques (connections). Including vertex merging techniques diminishes the negative effects of using independent camera subpaths in the MCMC sampler.

materials. Since such paths can be effectively sampled only by unidirectional path tracing technique, our algorithm provides no advantage over existing solutions in handling them (see Figure 6.23).

Future Work. In our algorithm, we map each primary sample to a set of full paths using all the VCM/UPS path sampling techniques. Hachisuka et al. [2014] have shown that using only one selected technique per primary sample can be more efficient (see Section 4.2.2). Using a similar approach in our algorithm could potentially improve its local exploration (however, it may come at the cost of worse global exploration, see Figure 5.16). Another interesting topic of future work would be robust rendering of participating media. For example, our approach could serve as a guideline for applying MCMC sampling to the work of Křivánek et al. [2014]. Furthermore, claiming that our work completely resolves the problem of unpredictable convergence behavior of MCMC would be an overstatement and there is certainly more work to be done on this topic.

6.6 Conclusion

We proposed a light transport simulation algorithm which fuses the concepts behind VCM/UPS and MCMC into a practical solution for the first time. Our algorithm can efficiently handle complex glossy and specular transport and complex visibility by taking advantage of both VCM/UPS and MCMC. The design of our algorithm is not a mere combination of the two concepts in a trivial manner. We instead made a number of careful design decisions based on practical insights on the building blocks of the algorithm. More specifically, we designed our algorithm to retain the benefits of combined bidirectional estimators, extensive subpath reuse, and spatial relaxation from VCM/UPS while minimizing the correlation problems of MCMC. The numerical experiments demonstrate that our algorithm indeed features the advantages of both VCM/UPS and MCMC, while suppressing their disadvantages in various scene configurations. Since our algorithm has low overhead over VCM/UPS and primary sample space MLT, it

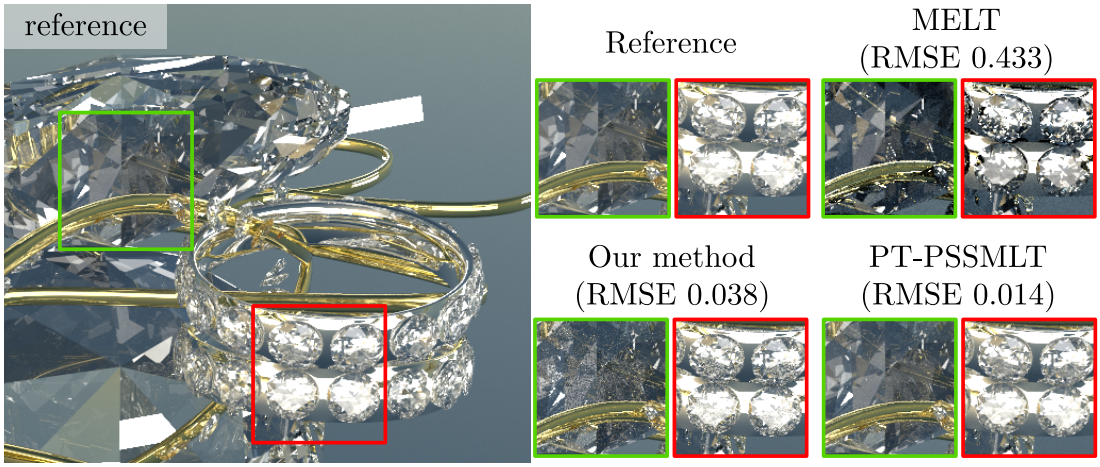

Figure 6.23: An equal-time comparison (10 minutes) of rendering a scene with only purely specular materials. While our algorithm delivers worse results than primary sample space MLT (see Section 4.2.1) over unidirectional path-tracing (PT-PSSMLT), it outperforms manifold exploration light transport (MELT, see Section 4.3.1), which gets stuck at local maxima of its target function.

can practically substitute those algorithms in many applications and improve the robustness of light transport simulation.

Appendix

6.A Algorithm components comparison

Fig. 6.24 shows images rendered by the different versions of our algorithm (see Fig. 6.21 for the description of the different versions).

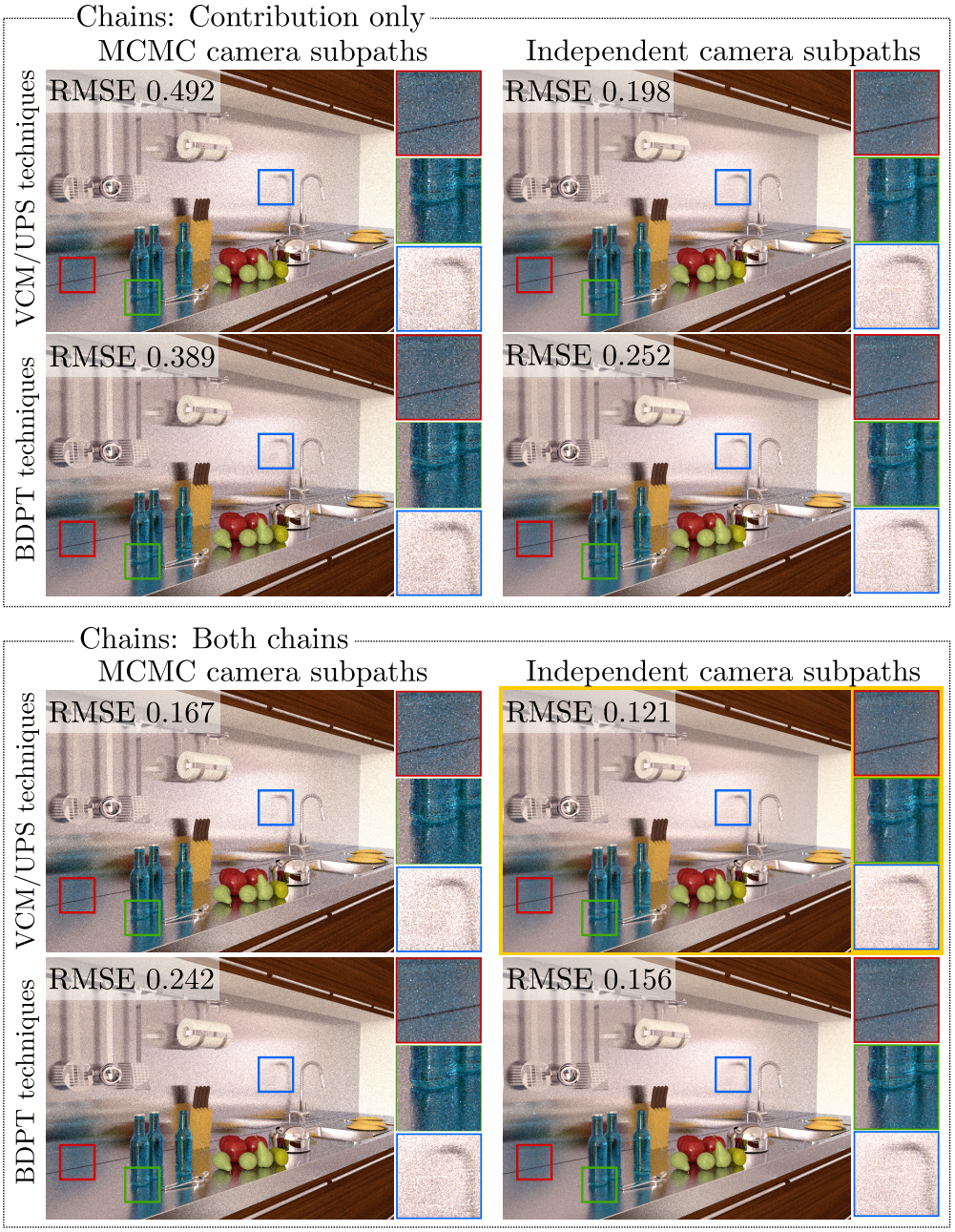

Figure 6.24: An equal-time (one hour) comparison of different variants of our algorithm. The final most robust method is marked by the orange rectangle.

6.B Mutating samples in the primary sample space

We base our algorithm on primary sample space MLT (PSSMLT, see Section 4.2.1). Our algorithm therefore applies mutations on a vector of random numbers (primary space samples) $\overline{u} \in \mathcal{P}$, rather than directly on paths. We use each $\overline{u}$ to generate a light subpath, by applying standard photon tracing technique and using $\overline{u}$ as input random numbers. The camera subpaths used in our algorithm do not rely on MCMC and are generated using standard Monte Carlo (for more information see Section 6.3).

In order to mutate a primary sample $\overline{u}$, we do not apply the original method used in PSSMLT. Instead we use the adaptive mutation kernel introduced to light transport by Hachisuka and Jensen [2011] (see Section 4.4.2). While we observed that our algorithm seems to generate noiseless images faster using the original PSSMLT's mutation kernel, it is also more prone to high sample correlation and the Markov chain also tends to get stuck at a local maxima of the target function more often, which results in fireflies and sometimes even non-uniform convergence. Therefore, we have chosen the more robust adaptive mutation kernel.

The adaptive mutation of $\overline{u}_{i+1}$ given a previous primary sample $\overline{u}_i$ is defined as follows

$$\overline{u}_{i+1} = \overline{u}_i + \begin{cases} (2\xi)^{1/\theta_i} & \text{if } \xi < 0.5 \\ -(2\xi - 1)^{1/\theta_i} & \text{otherwise.} \end{cases} \tag{6.8}$$

Here ξ is a uniform random number within $[0; 1)$ and θ_i is the adaptation parameter. The adaptation parameter θ_i is updated after $\overline{u}_{i+1}$ has been accepted or rejected by the MCMC algorithm using Equation (4.10).

6.C Our quest for a robust combination of MCMC and VCM/UPS

Even though our solution to combining MCMC and VCM/UPS may seem almost straightforward, we have in fact reached it after many failed attempts. We present here how we arrived to the final solution and we briefly mention some of the more promising alternatives we considered along the way.

6.C.1 Straightforward combination of PSSMLT and VCM/UPS

We began our attempts by searching for the most easy to implement and yet effective combination of MCMC and VCM/UPS. Since PSSMLT (see Section 4.2.1) is built on top of bidirectional path tracing (see Section 2.5), which only lacks the merging techniques compared to VCM/UPS, PSSMLT with added merging techniques (extended PSSMLT) seemed as a good starting point. Adding merging to PSSMLT was quite straightforward until we wanted to make it efficient by utilizing brute force path reuse (i.e. one subpath is merged with many others). Enabling brute force path reuse proved to be a major challenge that shaped the whole algorithm.

Generating subpaths for brute force path reuse

In the usual implementation of VCM/UPS, a full set of light subpaths is generated prior to creating any camera subpath in order to enable brute force path reuse. This is however not applicable in PSSMLT, since it uses MCMC to always generate a pair of camera and light subpath (which form full paths using connection techniques). A light subpath cannot be generated by PSSMLT without a camera subpath and vice versa. We solved this problem by using a separate set of light subpaths $\mathcal{L}$ used solely for merging. The MCMC algorithm used in our extended PSSMLT then looks as follows:

1. Propose a pair of camera and light subpath
2. Connect the pair of subpaths using connection techniques
3. Merge vertices of the camera subpath with all light subpaths in the set $\mathcal{L}$
4. Accept/reject the pair of subpaths given the total contribution

To obtain the required set of light subpath $\mathcal{L}$, we came up with two different methods:

Method A: First, we run standard PSSMLT and store light subpaths that it generates into $\mathcal{L}$. Later on, when $\mathcal{L}$ is large enough, we can switch to the extended PSSMLT and utilize light subpaths from $\mathcal{L}$ for merging. We further use light subpaths generated by MCMC to update $\mathcal{L}$.

Method B: The second solution is to generate $\mathcal{L}$ by an independent algorithm specialized at light subpath tracing, such as the visibility-driven photon tracing of Hachisuka and Jensen (see Section 4.4.2). The whole algorithm then iterates over two steps: first we generate $\mathcal{L}$ and then we run extended PSSMLT with given number of samples using $\mathcal{L}$ for merging.

Unfortunately, none of the above methods proved to be an effective solution. We discuss problems of each method in the following text.

Method A - computing light subpath probability

In order to compute probability of the merging technique (see Section 2.7), we need to know the probability with which light subpaths in $\mathcal{L}$ have been generated. In the method A we use previous iterations of PSSMLT to generate $\mathcal{L}$, therefore we must compute the probability with which PSSMLT generates a given light subpath. Unfortunately, as we discuss in detail in Section 6.3.4, computing analytically such probability is impossible due to PSSMLT's complicated target function. Implementing the extended PSSMLT with method A is therefore infeasible.

Method B - suboptimal MIS weights

In the method B, we used an independent MCMC algorithm with a simple visibility target function to generate $\mathcal{L}$, therefore we were able compute the required

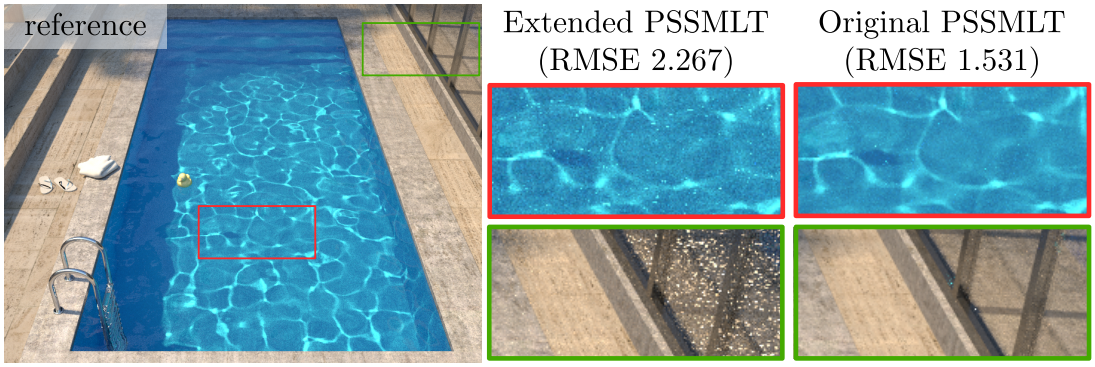

Figure 6.25: An equal-time comparison of the original PSSMLT and our extended version that uses visibility driven MCMC to generate light subpaths for merging. Due to the suboptimal combination of sampling techniques, our extended version yields worse results even for areas dominated by specular-diffuse-specular paths (these are usually handled best by the merging techniques).

probabilities (see Section 6.3.4). While this time we could implement the extended PSSMLT with all VCM/UPS path sampling techniques, the results were unsatisfactory. We can observe in Figure 6.25 that the extended PSSMLT is often less effective than the original PSSMLT.

The cause of this is that we used multiple importance sampling (MIS) weights (see Section 2.3) from VCM/UPS. We completely ignored the fact that merging and connection techniques use light subpaths generated by two different algorithms (visibility driven MCMC generates $\mathcal{L}$, while PSSMLT generates light subpaths for connections). In order to optimally combine the techniques we needed to project the probabilities with which these MCMC algorithms generated light subpaths into MIS weights. However, we already know that for light subpaths generated by PSSMLT we cannot compute the required probabilities. Thus we were forced to abandon the extended PSSMLT and use a different approach.

6.C.2 Splitting generation of subpaths

First, to solve the problems of the method B, we decided to split the generation of light subpaths and camera subpaths into two independent MCMC algorithms. This way, we could use the same set of light subpaths for both merging and connection techniques. Furthermore, to effectively compute subpath probability we only used a visibility target function. The proposed algorithm worked as follows:

1. Generate camera subpaths using visibility-driven MCMC
2. Generate light subpaths using visibility-driven MCMC
3. Combine the two sets of subpaths using VCM/UPS techniques
4. Iterate the above steps until convergence

While we solved the problems of the extended PSSMLT, we were facing a new problem, the definition of visibility for camera subpaths.

Visibility definition

Original visibility-driven photon tracing accepts a proposed light subpath only if it contributes to the image (i.e. if it can be merged with or connected to any camera subpath). We applied this for generation of our light subpaths as well. We also wanted to use MCMC to generate only camera subpaths that contribute to the image. Unfortunately, when the algorithm above generates camera subpaths, it has not yet generated light subpaths and therefore we could not estimate the contribution to the image.

We solved this problem by defining the visibility for camera subpaths using *old* light subpaths generated in the previous iterations. More specifically, an camera subpath is visible if any of its vertices lie in the vicinity of any vertex from old light subpaths (i.e. it can be merged with any light vertex). While this approach worked, the camera subpaths were often generated in places where they could not reach any light subpath from the current iteration. Due to this fact, the resulting algorithm often suffered from non-uniform convergence.

6.C.3 Final algorithm

Since the MCMC used to generate camera subpaths was the source of problems, we decided to replace it with a simple stratified Monte Carlo generator. Section 6.2 discusses that this decision has brought many advantages and resulted in a more effective algorithm. At this point our algorithm was already quite robust and performed better than state-of-the-art algorithms on our test scenes. However, we wanted to push it even further.

We noticed that for light subpaths the visibility target function is not always optimal, since light subpaths are often distributed in visible areas with low contribution to the image. Therefore, we experimented with different target functions, including the original PSSMLT contribution target function. Even though we had to settle with less optimal MIS weights (see Section 6.3.4) due to the use of MCMC with the contribution target function, the results were often better than with the visibility target function. To utilize the advantages of both target functions, we have combined them using replica exchange (see Section 4.5). Replica exchange was then the final piece of our new robust algorithm that combines the strengths of MCMC and VCM/UPS.

Conclusion

7.1 Summary

In this thesis we tackled a fundamental issue of light transport algorithms based on Markov chain Monte Carlo: insufficient global exploration. We proposed two different solutions that significantly improve global exploration in MCMC algorithms in scenes with difficult light transport.

In Chapter 5, we proposed a systematic attempt at improving global exploration using the replica exchange technique. To utilize full potential of replica exchange, we have addressed both of its key components: tempering and replica exchange moves. We showed that tempering light transport simulation by roughening BSDFs at vertices connecting subpaths from the camera and a light source simplifies the exploration of the state space by a Markov chain. We developed new replica exchange moves motivated by the inefficiency of the existing ones. We compared the different replica exchange moves with the equi-energy sampler [Kou et al., 2006], that was never used in light transport simulation before. While the resulting MCMC algorithm achieved better global exploration than existing methods, there is still a large space for improvement.

In Chapter 6, we proposed a different MCMC algorithm that improves global exploration by utilizing all path sampling techniques from the vertex connection and merging [Georgiev et al., 2012], a.k.a. unified path space sampling [Hachisuka et al., 2012] (VCM/UPS) algorithm. Due to the use of path sampling techniques with spatial regularization and subpath reuse, the proposed algorithm can efficiently explore the path space even in scenes with difficult light transport. To further improve global exploration, we used replica exchange with two chains: one based solely on path visibility and the other on the path contribution. Furthermore, we reduced correlation in the image space and improved image-plane stratification by decoupling the sampling of light subpaths (i.e. subpaths from a light source) and camera subpaths (i.e. subpaths from the camera), and by sampling the camera subpaths by ordinary (independent) Monte Carlo. The resulting algorithm not only achieves better global exploration than existing methods, but its convergence in the tested scenes is predictable, and thus it can be utilized for fast preview of the rendering and also for generating animations without temporal artifacts.

We believe that especially the algorithm introduced in Chapter 6 can find its application in practice, where it could be utilized in architectural visualization or the movie industry, to render scenes featuring both complex visibility and complex light transport. We also believe that our work will encourage further research into improving global exploration of the Markov chain Monte Carlo algorithms. We hope that such a research will allow the MCMC algorithms to be utilized in widely used rendering systems.

7.2 Future directions for improving global exploration

One of the reasons why the current MCMC algorithms still suffer from insufficient exploration is that they often achieve the exploration by employing completely uninformed uniform mutations (the large step mutation used in the primary sample space is at least locally informed due to the inherent importance sampling, see Section 4.2.1). While employing replica exchange or reducing the variation of the target function by utilizing various path sampling techniques certainly improves global exploration, in both cases the exploration still relies on these uniform mutations.

In order to utilize more informed mutations for global exploration, one could apply adaptive MCMC [Haario et al., 2001], that adapts mutations based on the past samples. While adaptive MCMC was already used in light transport simulation (and we utilize it in one of our algorithms as well, see Section 6.3.2), so far it was applied to only globally adapt the step size of local exploration mutations (e.g. small step in primary sample space MLT). We believe that one could learn from the past samples more information and adapt the exploration mutations in such a way that they would ensure frequent sampling of important parts of the path space as well as exploration of so-far unexplored parts.

Another option is to learn the mutation for global exploration by utilizing a pre-process. For example, the *path guiding* work by Vorba et al. [2014] learns distributions for sampling ray-tracing directions in ordinary Monte Carlo algorithms from an on-line stream of particles traced from light sources and the camera. This method can be utilized directly in the primary sample space algorithms, to improve the mapping from the primary sample space to the path space and thus allowing the large step mutation to use this global information. The information learned from the particles could also be used to adapt the mutations themselves.

7.3 Adopting Markov chain Monte Carlo in practice

While insufficient global exploration is one of the major obstacles in adopting Markov chain Monte Carlo light transport algorithms in practice, it is not the only one. Here we briefly mention two other important obstacles that must be overcome, if MCMC algorithms are to be used in practice.

In a production renderer it is often desirable to allow the users to influence the rendering in a non-physical manner. For example the user might want to illuminate a given object in the scene by a given light source only if it is visible from the camera either directly or through specular reflections/refractions. Another example of non-physical behavior often used in practice is a glass pane that refracts light only for some types of paths (pure specular ones), while for other light paths it allows the light to pass through without refraction. Such non-physical properties of objects are very useful in practice, however they are often difficult to implement, especially when the underlying light transport algorithm can generate the same light paths in various ways. Note that most of the cur-

rent MCMC algorithms either utilize multiple path sampling techniques to map samples to the path space or generate paths using various path space mutations. Implementing these non-physical controls is thus difficult for the current MCMC algorithms. The only exception is the primary sample space MLT built on top of a path tracer, which is however quite inefficient in scenes featuring complex light transport.

More robust light transport algorithms often have a significant overhead in many production scenes that can be easily rendered by a path tracer. While the original Metropolis light transport can render some simple scenes even faster than a path tracer, the more robust MCMC algorithms also have substantial overhead. Experienced users can certainly decide when a simpler light transport algorithm is sufficient to render a given scene, but for the most users it is preferential to have an algorithm that can perform well in both simple and complex scenes.

In conclusion, if a Markov chain Monte Carlo algorithm is truly to succeed in practice, the algorithm must be simple enough to enable all the non-physical controls, it must have low overhead and yet it must handle complex scenes as well, otherwise there is no advantage over a simple path tracer. Of course, such an algorithm must have excellent global exploration, so it can generate images progressively without artifacts. We hope that our work has made at least a small, yet not insignificant contribution toward this goal.

Bibliography

Christophe Andrieu and Christian P. Robert. Controlled MCMC for optimal sampling. Technical Report 0125, Cahiers de Mathématiques du Ceremade, Université Paris-Dauphine, 2001.

Christophe Andrieu and Johannes Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008. ISSN 1573-1375.

Christophe Andrieu, Ajay Jasra, Arnaud Doucet, and Pierre D. Moral. A note on convergence of the equi-energy sampler. *Stochastic Analysis and Applications*, 26(2):298–312, 2008.

Michael Ashikhmin, Simon Premože, Peter Shirley, and Brian Smits. A variance analysis of the Metropolis light transport algorithm. *Computers & Graphics*, 25(2):287 – 294, 2001. ISSN 0097-8493.

Yves F. Atchadé and Jun S. Liu. Discussion of "Equi-energy sampler" by Kou, Zhou and Wong. *The Annals of Statistics*, 34(4):1620–1628, 08 2006.

Yves F. Atchadé, Gareth O. Roberts, and Jeffrey S. Rosenthal. Towards optimal scaling of Metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, 21(4):555–568, 2010. ISSN 1573-1375.

Meïli Baragatti, Agnès Grimaud, and Denys Pommeret. Parallel tempering with equi-energy moves. *Statistics and Computing*, 23(3), 2012. ISSN 1573-1375.

Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. Reversible jump Metropolis light transport using inverse mappings. *ACM Transactions on Graphics*, 37(1):1:1–1:12, October 2017. ISSN 0730-0301.

David Blackwell. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, 18(1):105–110, 03 1947.

Steve Brooks, Andrew Gelman, Galin L. Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

Jiating Chen, Bin Wang, and Jun-Hai Yong. Improved stochastic progressive photon mapping with Metropolis sampling. *Computer Graphics Forum (EGSR 2011)*, 30(4):1205–1213, 2011.

Per H. Christensen. Adjoints and importance in rendering: An overview. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):329–340, 2003. ISSN 1077-2626.

Per H. Christensen and Wojciech Jarosz. The path to path-traced movies. *Foundations and Trends in Computer Graphics and Vision*, 10(2):103–175, October 2016. ISSN 1572-2740.

David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3):1186–1195, July 2005. ISSN 0730-0301.

Radu V. Craiu, Jeffrey Rosenthal, and Chao Yang. Learn from thy neighbor: Parallel-chain and regional adaptive MCMC. *Journal of the American Statistical Association*, 104(488):1454–1466, 2009.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216 – 222, 1987. ISSN 0370-2693.

Philip Dutré and Yves D. Willems. Importance-driven Monte Carlo light tracing. In Georgios Sakas, Stefan Müller, and Peter Shirley, editors, *Photorealistic Rendering Techniques*, pages 188–197. Springer Berlin Heidelberg, 1995. ISBN 978-3-642-87825-1.

David J. Earl and Michael W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7:3910–3916, 2005.

Shaohua Fan, Stephen Chenney, and Yu-chi Lai. Metropolis photon sampling with optional user guidance. In *Eurographics Symposium on Rendering (EGSR '05)*, pages 127–138, 2005.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (SIGGRAPH Asia '12)*, 31(6):192:1–192:10, 2012. ISSN 0730-0301.

Charles J. Geyer. Markov Chain Monte Carlo Maximum Likelihood. In *In Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 156–163. Defense Technical Information Center, 1991.

Adrien Gruson, Mickaël Ribardière, Martin Šik, Jiří Vorba, Rémi Cozot, Kadi Bouatouch, and Jaroslav Křivánek. A spatial target function for Metropolis photon tracing. *ACM Transactions on Graphics*, 36(1):4:1–4:13, November 2016. ISSN 0730-0301.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001.

Toshiya Hachisuka and Henrik W. Jensen. Stochastic progressive photon mapping. *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, 28(5):141:1–141:8, 2009.

Toshiya Hachisuka and Henrik W. Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics*, 30(5):114:1–114:11, 2011. ISSN 0730-0301.

Toshiya Hachisuka, Shinji Ogaki, and Henrik W. Jensen. Progressive photon mapping. *ACM Transactions on Graphics (SIGGRAPH Asia '08)*, 27(5):130:1–130:8, 2008. ISSN 0730-0301.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik W. Jensen. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (SIGGRAPH Asia '12)*, 31(6):191:1–191:10, 2012. ISSN 0730-0301.

Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. Multiplexed Metropolis light transport. *ACM Transactions on Graphics*, 33(4): 100:1–100:10, July 2014. ISSN 0730-0301.

Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. Improved half vector space light transport. *Computer Graphics Forum*, 34(4):65–74, 2015.

Wilfred K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444.

Jared Hoberock and John C. Hart. Arbitrary importance functions for Metropolis light transport. *Computer Graphics Forum*, 29(6):1993–2003, 2010.

Xia Hua and S. C. Kou. Convergence of the equi-energy sampler and its application to the ising model. *Statistica Sinica*, 21(4):1687–1711, 2011. ISSN 10170405, 19968507.

Wenzel Jakob. Mitsuba renderer, 2010. http://mitsuba-renderer.org.

Wenzel Jakob and Steve Marschner. Manifold exploration: A Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics*, 31(4):58:1–58:13, 2012. ISSN 0730-0301.

Wenzel Jakob and Steve Marschner. Path space MCMC methods in computer graphics. In Ronald Cools and Dirk Nuyens, editors, *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014 (Springer Proceedings in Mathematics & Statistics)*. Springer International Publishing, July 2016.

Henrik W. Jensen. Importance driven path tracing using the photon map. In *Eurographics Workshop Rendering*, pages 326–335, 1995.

Henrik W. Jensen. Global illumination using photon maps. In *Eurographics Workshop on Rendering*, pages 21–30. Springer-Verlag, 1996.

James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4): 143–150, August 1986. ISSN 0097-8930.

Anton S. Kaplanyan and Carsten Dachsbacher. Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Eurographics 2013)*, 32(2):63–72, 2013.

Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics*, 33(4):102:1–102:13, July 2014. ISSN 0730-0301.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum (Eurographics)*, 21(3):531–540, 2002. ISSN 1467-8659.

Alexander Keller. Instant Radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 49–56, 1997. ISBN 0-89791-896-7.

Shinya Kitaoka, Yoshifumi Kitamura, and Fumio Kishino. Replica exchange light transport. *Computer Graphics Forum*, 28(8):2330–2342, 2009. ISSN 1467-8659.

S. C. Kou, Qing Zhou, and Wing H. Wong. Equi-energy sampler with applications in statistical inference and statistical mechanics. *The Annals of Statistics*, 34 (4):1581–1619, 08 2006.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (SIGGRAPH 2014)*, 33(4):103:1–103:13, July 2014. ISSN 0730-0301.

Jaroslav Křivánek, Alexander Keller, Iliyan Georgiev, Anton Kaplanyan, Marcos Fajardo, Mark Meyer, Jean-Daniel Nahmias, Ondřej Karlík, and Juan Canada. Recent advances in light transport simulation: Some theory and a lot of practice. In *ACM SIGGRAPH 2014 Courses*, 2014.

Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Compugraphics '93*, pages 145–153, 1993.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain Metropolis light transport. *ACM Transactions on Graphics (SIGGRAPH 2013)*, pages 95:1–95:12, July 2013. ISSN 0730-0301.

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Transactions on Graphics (SIGGRAPH Asia 2015)*, 34(6):209:1–209:13, 2015. ISSN 0730-0301.

Jun S. Liu, Faming Liang, and Wing Hung Wong. The Multiple-Try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000. ISSN 01621459.

Vasilios I. Manousiouthakis and Michael W. Deem. Strict detailed balance is unnecessary in Monte Carlo simulation. *The Journal of Chemical Physics*, 110 (6):2753–2756, 1999.

Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. Improved sampling for gradient-domain Metropolis light transport. *ACM Transactions on Graphics*, 33(6):178:1–178:12, November 2014. ISSN 0730-0301.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Błażej Miasojedow, Eric Moulines, and Matti Vihola. An adaptive parallel tempering algorithm. *Journal of Computational and Graphical Statistics*, 22(3): 649–664, 2013.

Antonietta Mira. On Metropolis-Hastings algorithms with delayed rejection. *Metron - International Journal of Statistics*, 0(3-4):231–241, 2001.

Radford M. Neal. *Handbook of Markov Chain Monte Carlo*, chapter MCMC using Hamiltonian dynamics. CRC press, 2011.

Esa Nummelin. *General irreducible Markov chains and non-negative operators*. Cambridge University Press, 1984.

Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. Fusing state spaces for Markov chain Monte Carlo rendering. *ACM Transactions on Graphics*, 36(4):74:1–74:10, July 2017. ISSN 0730-0301.

Jacopo Pantaleoni. Charted Metropolis light transport. *ACM Transactions on Graphics*, 36(4):75:1–75:14, July 2017. ISSN 0730-0301.

Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 11–22, 2000. ISBN 3-211-83535-0.

Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, July 2003. ISSN 0730-0301.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering (Third Edition)*. Morgan Kaufmann, Boston, 2017. ISBN 978-0-12-800645-0.

Dan Piponi. Automatic differentiation, c++ templates, and photogrammetry. *J. Graphics, GPU, & Game Tools*, 9:41–55, 2004.

Pixar. RenderMan, 1982. https://renderman.pixar.com/.

Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. Probabilistic connections for bidirectional path tracing. *Computer Graphics Forum*, 34(4):75–86, 2015.

Jeffrey S. Rosenthal. *Optimal Proposal Distributions and Adaptive MCMC*, chapter 4. Handbooks of Modern Statistical Methods. Chapman & Hall, CRC, Florida, USA, first edition, 2011.

Martin Schmidt, Oleg Lobachev, and Michael Guthe. Coherent Metropolis light transport on the GPU using speculative mutations. In *Journal of WSCG. 2016, vol. 24*, number 1 in Journal of WSCG. 2016, pages 1–8, 2016.

Amandine Schreck, Gersende Fort, and Eric Moulines. Adaptive equi-energy sampler: Convergence and illustration. *ACM Trans. Model. Comput. Simul.*, 23(1):5:1–5:27, January 2013. ISSN 1049-3301.

Benjamin Segovia, Jean-Claude Iehl, and Bernard Péroche. Metropolis instant radiosity. *Computer Graphics Forum (Eurographics 2007)*, 26(3):425–434, 2007.

Benjamin Segovia, Jean-Claude Iehl, and Bernard Péroche. Coherent Metropolis light transport with Multiple-Try mutations. Technical Report RR-LIRIS-2007-015, LIRIS UMR 5205 CNRS/INSA de Lyon, April 2007.

Hidemaro Suwa and Synge Todo. Markov chain Monte Carlo method without detailed balance. *Physical Review Letters*, 105:120603, Sep 2010.

Robert H. Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57:2607–2609, 1986.

László Szirmay-Kalos, Péter Dornbach, and Werner Purgathofer. On the startup bias problem of Metropolis sampling. *Journal of WSCG*, 7(1-2):273–280, 1999.

László Szirmay-Kalos, Balázs Benedek, and Mateu Sbert. Metropolis iteration for global illumination. *Journal of WSCG*, 12(1-3):451–458, 2004.

Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994. ISSN 00905364.

Eric Veach. *Robust Monte Carlo methods for light transport simulation.* PhD thesis, Stanford University, 1997.

Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Proc. Eurographics Rendering Workshop*, pages 147–162, 1994.

Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 419–428, 1995. ISBN 0-89791-701-4.

Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, 1997. ISBN 0-89791-896-7.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics*, 33(4):101:1–101:11, July 2014. ISSN 0730-0301.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 195–206, Aire-la-Ville, Switzerland, Switzerland, 2007. ISBN 978-3-905673-52-4.

Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 265–272, New York, NY, USA, 1992. ACM. ISBN 0-89791-479-1.

Quan Zheng and Chang-Wen Zheng. Visual importance-based adaptive photon tracing. *The Visual Computer*, 31(6-8):1001–1010, June 2015. ISSN 0178-2789.

Károly Zsolnai-Fehér and László Szirmay-Kalos. Automatic parameter control for Metropolis light transport. In *EG 2013 – Short Papers*, pages 53–56, 2013.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proceedings of Eurographics)*, 34(2):667–681, May 2015.

List of Abbreviations

MCMC	Markov chain Monte Carlo
pdf	probability density function
BSDF	bidirectional scattering distribution function
MIS	multiple importance sampling
BDPT	bidirectional path tracing
SPPM	stochastic progressive photon mapping
VCM/UPS	vertex connection and merging/unified path sampling
MH	Metropolis-Hastings
MC	Monte Carlo
MLT	Metropolis light transport
PSSMLT	primary sample space Metropolis light transport
MMLT	multiplexed Metropolis light transport
RR	Russian roulette
MELT	manifold exploration light transport
HSLT	half-vector space light transport
NCS	natural constraints space
HMC	Hamiltonian Monte Carlo
H^{2}MC	Hessian-Hamiltonian Monte Carlo
AMCMC	adaptive Markov chain Monte Carlo
RELT	replica exchange light transport
MTM	multiple-try Metropolis
MIR	Metropolis instant radiosity
IR	instant radiosity
VPL	virtual point light
GDMLT	gradient domain Metropolis light transport
ERPT	energy redistribution path tracing
EEM	equi-energy moves
EES	equi-energy sampler
IS	importance-sampled
SDS	specular-diffuse-specular
RMSE	root mean square error
EV	exposure value
ERPT+ME	energy redistribution path tracing with manifold exploration

Attachments

Project Pages

- Robust light transport simulation via Metropolised bidirectional estimators: `http://cgg.mff.cuni.cz/~sik/meb/index.html`

Supplemental Images

- Robust light transport simulation via Metropolised bidirectional estimators: `http://cgg.mff.cuni.cz/~sik/meb/files/comparisons/index.html`