Langevin Monte Carlo Rendering with Gradient-based Adaptation

FUJUN LUAN, Cornell University
SHUANG ZHAO, University of California, Irvine
KAVITA BALA, Cornell University
IOANNIS GKIOULEKAS, Carnegie Mellon University

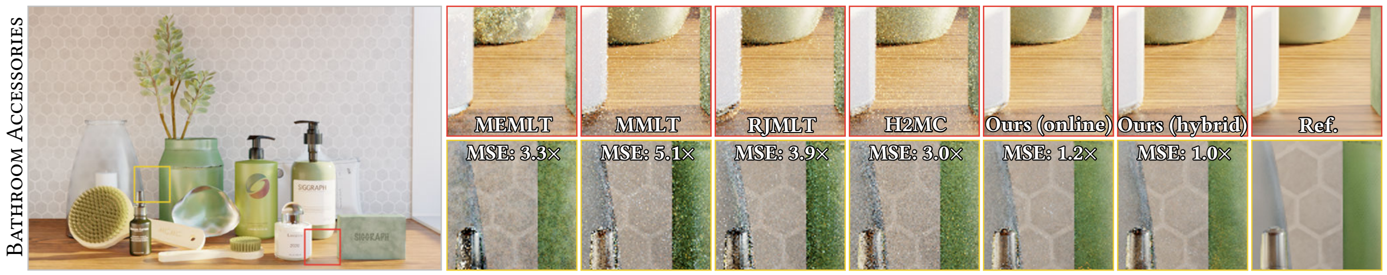

Fig. 1. Equal-time (20 minutes) comparison between MEMLT, MMLT, RJMLT, H2MC and two variants of our methods. The scene presents complex glossy and specular interreflections with difficult visibilities. The reference (Ref.) is rendered by BDPT in roughly a day. MEMLT suffers from correlated noise in the glass bottle region due to insufficient local exploration. MMLT and RJMLT both have severe noise because they sometimes get trapped in regions with hard-to-find features using Kelemen-style isotropic mutations. H2MC is more efficient with the help of anisotropic Gaussian mutations, but the computational overhead from Hessian computations and dense matrix operations results in a low sample budget and insufficient exploration. Our Langevin Monte Carlo methods efficiently address these challenges by exploiting first-order gradient information to robustly balance the tradeoff between adaptation and cost.

We introduce a suite of Langevin Monte Carlo algorithms for efficient photorealistic rendering of scenes with complex light transport effects, such as caustics, interreflections, and occlusions. Our algorithms operate in primary sample space, and use the Metropolis-adjusted Langevin algorithm (MALA) to generate new samples. Drawing inspiration from state-of-the-art stochastic gradient descent procedures, we combine MALA with adaptive preconditioning and momentum schemes that re-use previously-computed first-order gradients, either in an online or in a cache-driven fashion. This combination allows MALA to adapt to the local geometry of the primary sample space, without the computational overhead associated with previous Hessian-based adaptation algorithms. We use the theory of controlled Markov chain Monte Carlo to ensure that these combinations remain ergodic, and are therefore suitable for unbiased Monte Carlo rendering. Through extensive experiments, we show that our algorithms, MALA with online and cache-driven adaptation, can successfully handle complex light transport in a large variety of scenes, leading to improved performance (on average more than 3× variance reduction at equal time, and 7× for motion blur) compared to state-of-the-art Markov chain Monte Carlo rendering algorithms.

CCS Concepts: • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: global illumination, photorealistic rendering, Langevin Monte Carlo

Authors' addresses: Fujun Luan, Cornell University, fujun@cs.cornell.edu; Shuang Zhao, University of California, Irvine, shz@ics.uci.edu; Kavita Bala, Cornell University, kb@cs.cornell.edu; Ioannis Gkioulekas, Carnegie Mellon University, igkioule@andrew.cmu.edu.

0730-0301/2020/7-ART140 $15.00
https://doi.org/10.1145/3386569.3392382

ACM Reference Format:
Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Langevin Monte Carlo Rendering with Gradient-based Adaptation. *ACM Trans. Graph.* 39, 4, Article 140 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392382

1 INTRODUCTION

The development of general-purpose and efficient global illumination algorithms is one of the foundational problems in computer graphics. Physics-based Monte Carlo rendering algorithms [Dutre et al. 2006; Pharr et al. 2016] can accurately simulate complicated light transport effects such as caustics, strong interreflections, subsurface scattering, and motion blur. They achieve this by aggregating intensity contributions from a large number of randomly generated *light paths*, representing the different ways in which light can propagate through the scene that is being simulated. As the complexity of the underlying transport effects increases, retaining efficiency requires using Markov chain Monte Carlo (MCMC) techniques [Veach and Guibas 1997]: unlike traditional Monte Carlo algorithms, which generate independent paths through local importance sampling, MCMC algorithms use Markov chains to create statistically-correlated paths. This allows them to efficiently perform both a global exploration of the space of possible paths, searching for clusters of paths with high contributions to the image, and a local exploration of such newly discovered clusters.

The success of MCMC algorithms depends critically on the design of proposal distributions for producing new path samples given previously sampled ones. Whereas so-called *path space* MCMC rendering algorithms directly modify the path within the scene, *primary sample space* algorithms [Kelemen et al. 2002] modify the random numbers provided as input to a black-box path tracing algorithm. Operating in the mathematically-tractable space of real random numbers provides great flexibility for designing proposal distributions, and allows incorporating general-purpose proposals that have found success in other application areas of MCMC inference. This includes state-of-the-art gradient-based proposal distributions such as Langevin Monte Carlo (LMC) [Roberts and Tweedie 1996] and Hamiltonian Monte Carlo (HMC) [Duane et al. 1987]: essentially, these work by performing one (LMC) or multiple (HMC) steps of a noisy gradient ascent procedure, that guides the sampling process towards parts of the primary sample space corresponding to local maxima of the path contribution function.

The use of gradient-based MCMC techniques for physics-based rendering has been hindered by two main factors: First, generating path samples requires the ability to differentiate complicated path-tracing algorithms, corresponding to sequences of reflection, transmission, and volume scattering events. Second, producing high-quality samples requires additional expensive computation, such as computing, factorizing, and inverting the Hessian matrix of the path contribution function for each new path sample [Li et al. 2015]. Analogous to the use of the Hessian in Newton's method, this helps the gradient ascent procedure underlying the sampling process adapt to the local geometry of the primary sampling space.

The recent proliferation of rendering engines that support end-to-end differentiation [Anderson et al. 2017; Che et al. 2018; Li et al. 2018; Nimier-David et al. 2019; Zhang et al. 2019] helps overcome the first of these two factors, and motivates us to revisit the use of gradient-based MCMC for physics-based rendering. Our focus is on developing proposal distributions based on LMC, coupled with adaptation mechanisms to improve the exploration of the primary sampling space. To make these LMC with adaptation processes efficient enough for rendering, we make the following contributions:

- Inspired by state-of-the-art stochastic gradient descent (SGD) algorithms [Kingma and Ba 2014], we propose adaptation mechanisms that introduce minimal computational overhead compared to standard LMC. These include a Hessian approximation and a momentum vector that can be computed by performing only scalar operations on first-order gradients, sidestepping the need for expensive second-order differentiation and matrix operations.
- We combine these adaptation mechanisms with a new caching scheme, that stores previously computed gradients, and uses them to guide the sampling process in a way that facilitates both local and global exploration.
- We use theoretical results on *controlled MCMC processes* [Andrieu and Thoms 2008], to ensure the ergodicity and correctness of these combinations.

These contributions result in three MCMC rendering algorithms that significantly outperform state-of-the-art techniques, resulting in an average MSE improvement of 3× on equal-time comparisons across a large variety of challenging scenes, and 7× for scenes with motion blur. We make our implementation publicly-available [Luan et al. 2020], to facilitate reproducibility and follow-up research.

2 RELATED WORK

Gradient-based and controlled Markov chain Monte Carlo. Markov chain Monte Carlo (MCMC) refers to a large class of techniques that allow sampling from arbitrary distributions (known only up to scale) to perform various statistical inference tasks (for example, computing expectations) [Brooks et al. 2011]. Perhaps the most famous among these techniques is the Metropolis-Hastings algorithm [Hastings 1970; Metropolis et al. 1953], which uses an auxiliary *proposal* distribution to create a Markov chain that converges to the target distribution. State-of-the-art techniques for designing proposal distributions include *Hamiltonian Monte Carlo* (HMC) [Betancourt 2017; Duane et al. 1987; Neal et al. 2011] and *Langevin Monte Carlo* (LMC) [Roberts and Tweedie 1996], which generate proposals by simulating Hamiltonian and Langevin dynamics, respectively.

Both HMC and LMC use gradients of the target distribution, to steer the Markov chain towards places where the target distribution takes large values. Their similarity to gradient-descent optimization algorithms such as Adagrad [Duchi et al. 2011] and Adam [Kingma and Ba 2014] has inspired modifications that allow HMC and LMC to better adapt to the geometry of the sampling domain [Betancourt 2013; Girolami and Calderhead 2011], or move faster towards local maxima [Chen et al. 2016, 2014; Durmus et al. 2016; Ma et al. 2015; Simsekli et al. 2016; Welling and Teh 2011; Zhang and Sutton 2011]. The resulting proposal distributions often involve several parameters, requiring significant fine-tuning effort. Controlled MCMC techniques [Andrieu and Thoms 2008; Roberts and Rosenthal 2009] help alleviate this, by automatically adjusting parameters during the sampling process. We focus on a combination of optimization-inspired extensions to LMC with controlled MCMC, which we show can be successfully used for physics-based rendering.

MCMC rendering. MCMC techniques were introduced to physics-based rendering by Veach and Guibas [1997]. Their algorithm, termed Metropolis Light Transport, operates in *path space*, generating new path proposals by directly modifying the vertices of previously sampled paths. Subsequent path-space algorithms have introduced proposal strategies targeting hard-to-sample specular or near-specular paths [Hanika et al. 2015; Jakob and Marschner 2012; Kaplanyan et al. 2014], dealing with complex visibility [Otsu et al. 2018], or focusing on spatial image gradients [Lehtinen et al. 2013]. Alternatively, Kelemen et al. [2002] introduced MCMC rendering algorithms that operate in *primary sample space*, and generate new path proposals by modifying the random numbers used to generate previous paths. Our algorithms fall within this category, which also includes algorithms that consider multiple sampling strategies [Hachisuka et al. 2014], as well as incorporate path-space information [Bitterli et al. 2018; Otsu et al. 2017; Pantaleoni 2017], potentially only where necessary [Bitterli and Jarosz 2019]. MCMC path sampling has also found use within rendering algorithms based on density estimation techniques [Hachisuka and Jensen 2011; Šik et al. 2016].

Derivatives in rendering. Derivatives of quantities computed by rendering algorithms have been used in different ways to accelerate the rendering process. Classic approaches include ray differentials [Igehy 1999], path differentials [Suykens and Willems 2001], and irradiance gradients [Arvo 1994; Holzschuch and Sillion 1995; Jarosz et al. 2012, 2008b; Marco et al. 2018; Ramamoorthi et al. 2007; Schwarzhaupt et al. 2012; Ward and Heckbert 1992]. In path-space MCMC rendering, first-order derivatives have been used to perturb difficult to sample paths containing specular-diffuse-specular segments [Chen and Arvo 2000; Jakob and Marschner 2012]). Alternatively, Li et al. [Li et al. 2015] use both first and second-order derivatives, together with an HMC-inspired proposal distribution, to generate proposals in primary sample space. Our algorithms follow a similar approach, but use only first-order gradient information to generate high-quality proposals, without the need for expensive second-order differentiation. We discuss the relationship between our algorithm and that of Li et al. [2015] in more detail in Section 4. Finally, the proliferation of automatic differentiation techniques [Griewank and Walther 2008] in recent years has motivated the development of theory and systems for end-to-end differentiable rendering [Che et al. 2018; Li et al. 2018; Nimier-David et al. 2019; Zhang et al. 2019], which can facilitate not only faster rendering, but also *inverse* rendering [Azinović et al. 2019; Gkioulekas et al. 2016, 2013; Khungurn et al. 2015; Tsai et al. 2019; Zhao et al. 2016] and integration with learning-based pipelines [Che et al. 2018].

Caching in rendering. Using cached information generated during preprocessing or on the fly has a long history in rendering. Irradiance caching [Ward et al. 1988] accelerates the computation of indirect illumination by storing a sparse set of incident illumination samples, and interpolating between them at render time. This idea is improved by many follow-up works [Křivánek et al. 2006; Krivanek et al. 2005; Schwarzhaupt et al. 2012; Ward and Heckbert 1992], and generalized to handle participating media [Jarosz et al. 2008a; Marco et al. 2018]. Photon mapping [Jensen 2001] and its variants [Hachisuka and Jensen 2009; Jarosz et al. 2011; Křivánek et al. 2014] cache points or beam samples in photon maps, which they use with kernel density estimation at render time to handle complex light transport effects such as caustics. Alternatively, path-guiding methods [Jensen 1995; Müller et al. 2017; Müller et al. 2019; Reibold et al. 2019; Vorba et al. 2014; Zheng and Zwicker 2019] store incident radiance information during "training" phases, then use this information to better sample light paths. Compared to these prior works, we utilize caching in a fundamentally different way, by storing primary-sample-space gradient information used to improve the convergence of LMC. As this cached information only assists sample generation, and is not used to estimate path contributions, it does not affect the unbiasedness of our rendering algorithms.

3 MARKOV CHAIN MONTE CARLO RENDERING

We begin by introducing the problem setting, mathematical notation, and algorithmic concepts for Markov Chain Monte Carlo rendering. These will serve as background for developing our contributions.

Primary sample space. Our focus will be on physics-based Monte Carlo rendering algorithms that use the *primary sample space* formulation of light transport, first proposed by Kelemen et al. [2002]. This assumes that we have available a path generation algorithm, such as path tracing [Kajiya 1986], particle tracing [Arvo 1986; Dutré et al. 1993], or bidirectional path tracing (BDPT) [Lafortune and Willems 1993; Veach and Guibas 1995]. Such an algorithm deterministically transforms a sequence of $r(B)$ uniform random variables into a *light path* of length $B \in [2, \ldots \infty)$, that is, an ordered sequence $\bar{\mathbf{x}} = \mathbf{x}_1 \to \cdots \to \mathbf{x}_B$ of three-dimensional points on the surfaces or in the participating media that make up the scene. We can then define the space of all light paths as the *path space* $\mathcal{P}$, the union of hypercubes needed to generate all of the path space as the *primary sample space* $\mathcal{U} = \bigcup_{B=0}^{\infty} [0,1]^{r(B)}$, and the path generation algorithm as a map $S : \mathcal{U} \to \mathcal{P}$ between the two spaces.

With this notation at hand, Kelemen et al. [2002] proposed expressing radiometric measurements as integrals of the form:

$$I = \int_{\mathcal{U}} f\left(S\left(\bar{\mathbf{u}}\right)\right) \left|\frac{\mathrm{d}\mu\left(\bar{\mathbf{u}}\right)}{\mathrm{d}\bar{\mathbf{u}}}\right| \mathrm{d}\bar{\mathbf{u}} = \int_{\mathcal{U}} \tilde{f}\left(\bar{\mathbf{u}}\right)\, \mathrm{d}\bar{\mathbf{u}}, \tag{1}$$

where the *measurement contribution function* f describes the radiance that flows through the light path $\bar{\mathbf{x}} = S(\bar{\mathbf{u}})$, and $\tilde{f}(\bar{\mathbf{u}}) \equiv f(S(\bar{\mathbf{u}})) \left|\frac{\mathrm{d}\mu(\bar{\mathbf{u}})}{\mathrm{d}\bar{\mathbf{u}}}\right|$. This integral is a re-parameterization of the path-space integral formulation of light transport [Veach 1997; Veach and Guibas 1997], with the Jacobian term in Equation (1) accounting for the change of variables from path space to primary sample space. Under technical conditions that are typically satisfied by transformations S corresponding to standard path tracing algorithms, we can write $\tilde{f}(\bar{\mathbf{u}}) = f(\bar{\mathbf{x}})/p(\bar{\mathbf{x}})$, where $p(\bar{\mathbf{x}})$ is the probability density of the path $\bar{\mathbf{x}} = S(\bar{\mathbf{u}})$ [Anderson et al. 2017; Kelemen et al. 2002]. The primary sample space has been extended to account for multiple importance sampling [Hachisuka et al. 2014] and mixed-space techniques [Bitterli et al. 2018; Otsu et al. 2017; Pantaleoni 2017].

Monte Carlo rendering algorithms approximate I by aggregating weighted contributions from randomly generated sequences $\bar{\mathbf{u}}^{(t)}$:

$$\langle I \rangle \equiv \frac{1}{T} \sum_{t=1}^{T} \frac{\tilde{f}\left(\bar{\mathbf{u}}^{(t)}\right)}{\tilde{p}\left(\bar{\mathbf{u}}^{(t)}\right)}. \tag{2}$$

Typically, sampling in primary sample space is done using MCMC techniques, though independent sampling techniques also exist [Müller et al. 2019; Reibold et al. 2019; Zheng and Zwicker 2019].

Markov chain Monte Carlo rendering. MCMC rendering algorithms, first introduced by Veach and Guibas [1997], deviate from traditional Monte Carlo rendering in terms of how they generate samples for the estimator of Equation (2): instead of generating each sample independently, they use Markov chains to create *correlated sequences* of samples. In the following, we describe MCMC rendering in primary sample space [Kelemen et al. 2002].

Most commonly, MCMC rendering techniques are based on the Metropolis-Hastings algorithm [Hastings 1970; Metropolis et al. 1953]. This assumes that we have available a *proposal distribution* $\mathcal{T} : \mathcal{U} \times \mathcal{U} \to \mathbb{R}_{\geq}$. Given the current primary sample sequence $\bar{\mathbf{u}}^{(t)}$, we first sample a *proposed sequence* $\bar{\mathbf{v}} \sim \mathcal{T}\left(\bar{\mathbf{u}}^{(t)} \to \cdot\right)$.[1] Then, the proposed sequence is accepted with an *acceptance probability*:

$$\alpha\left(\bar{\mathbf{u}}^{(t)} \to \bar{\mathbf{v}}\right) = \min\left\{1, \frac{\tilde{f}\left(\bar{\mathbf{v}}\right) \mathcal{T}\left(\bar{\mathbf{v}} \to \bar{\mathbf{u}}^{(t)}\right)}{\tilde{f}\left(\bar{\mathbf{u}}^{(t)}\right) \mathcal{T}\left(\bar{\mathbf{u}}^{(t)} \to \bar{\mathbf{v}}\right)}\right\}. \tag{3}$$

If the proposed sequence is accepted, then the current sequence is updated as $\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{v}}$, otherwise $\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{u}}^{(t)}$. Each new sequence $\bar{\mathbf{u}}^{(t+1)}$ is used to update the Monte Carlo estimate of Equation (2), with a probability distribution $\tilde{p}\left(\bar{\mathbf{u}}^{(t+1)}\right) = \tilde{f}\left(\bar{\mathbf{u}}^{(t+1)}\right)/L$, where L is a constant estimated using independent Monte Carlo rendering.

A Monte Carlo estimate (Equation (2)) formed using sequences sampled with the above Markov chain procedure will be unbiased

[1]We denote by $\mathcal{T}(\cdot)$ the probability distribution a sample $\bar{\mathbf{v}}$ is drawn from, and by $\mathcal{T}(\bar{\mathbf{v}})$ the corresponding probability density function evaluated at $\bar{\mathbf{v}}$.

and consistent if the Markov chain satisfies two conditions: First, it has a stationary distribution that is proportional to the function $\tilde{f}$. Second, it is ergodic, meaning that it will converge to this stationary distribution from all initial sequences $\bar{\mathbf{u}}_0$. As a consequence of the use of the Metropolis-Hastings rule of Equation (3), both conditions will be satisfied if we select a proposal distribution that satisfies: $\mathcal{T}(\bar{\mathbf{u}} \to \bar{\mathbf{v}}) > 0$, for all $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ such that $\tilde{f}(\bar{\mathbf{u}}) > 0$ and $\tilde{f}(\bar{\mathbf{v}}) > 0$.

The selection of the proposal distribution $\mathcal{T}$ critically affects the performance of MCMC rendering techniques. As observed by Kelemen et al. [2002], it is important to combine two types of proposals: first, *large-step mutation* proposals $\mathcal{T}_{\text{global}}$ that can easily reach any sequence from any other sequence; second, *small-step perturbation* proposals $\mathcal{T}_{\text{local}}$ that generate a proposed sequence by only making small modifications to the previous sequence. Concretely:

$$\mathcal{T}(\bar{\mathbf{u}} \to \bar{\mathbf{v}}) = \pi_{\text{global}}\mathcal{T}_{\text{global}}(\bar{\mathbf{u}} \to \bar{\mathbf{v}}) + \pi_{\text{local}}\mathcal{T}_{\text{local}}(\bar{\mathbf{u}} \to \bar{\mathbf{v}}), \quad (4)$$

where $\pi_{\text{global}}, \pi_{\text{local}} > 0$, $\pi_{\text{global}} + \pi_{\text{local}} = 1$. Such combinations guarantee ergodicity, and can efficiently explore the target space, which typically consists of many disjointed subsets (informally termed "islands" by Kelemen et al. [2002]): The large-step mutation enables global exploration by jumping from one island to another, and the small-step perturbation enables local exploration by moving within one island. The large-step mutation proposal is typically implemented by uniformly sampling a sequence independently of the previous one: $\mathcal{T}_{\text{global}}(\bar{\mathbf{u}} \to \bar{\mathbf{v}}) = \mathcal{T}_{\text{global}}(\bar{\mathbf{v}})$. We use Langevin Mone Carlo to develop a new class of small-step perturbation proposals.

4 LANGEVIN MONTE CARLO

At the basis of our proposed rendering algorithms is a class of MCMC techniques known as *Langevin Monte Carlo* (LMC) [Roberts and Tweedie 1996]. Even though LMC techniques have become popular in other application areas of MCMC, they remain relatively unexplored in physics-based rendering—we are aware of one exception, the H2MC algorithm [Li et al. 2015], which however is derived from the related Hamiltonian Monte Carlo (HMC) [Betancourt 2017; Duane et al. 1987; Neal et al. 2011] framework. To keep our paper self-contained, we provide a brief review of LMC.

LMC techniques take their name from the *Langevin diffusion process* [Lemons and Gythiel 1997]. This is a continuous-time Markov process $\bar{\mathbf{u}}(t), t \in [0, \infty)$, that, given a positive function $\tilde{f} : \mathcal{U} \to \mathbb{R}_{\geq}$, satisfies the stochastic differential equation [Øksendal 2003]:

$$\mathrm{d}\bar{\mathbf{u}}(t) = \frac{1}{2}\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}(\bar{\mathbf{u}}(t))\,\mathrm{d}t + \mathrm{d}\bar{\mathbf{W}}(t). \quad (5)$$

The evolution of $\bar{\mathbf{u}}(t)$ is controlled by a deterministic *drift* term proportional to the gradient of $\tilde{\mathcal{L}}(\bar{\mathbf{u}}(t)) \equiv \log \tilde{f}(\bar{\mathbf{u}}(t))$, and the random Brownian motion term $\bar{\mathbf{W}}(t), t \in [0, \infty)$. This Markov process is ergodic, with a stationary distribution proportional to $\tilde{f}(\bar{\mathbf{u}}(t))$.

In practice, to simulate the Langevin diffusion, it is necessary to use a discrete approximation, such as the Euler-Maruyama discretization [Maruyama 1955]. Given a temporal step-size $\epsilon > 0$, this corresponds to the following discrete-time Markov chain:

$$\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right) + \sqrt{\epsilon}\bar{\mathbf{w}}, \quad (6)$$

where $\bar{\mathbf{w}} \sim \mathcal{N}(\cdot; \mathbf{0}, \mathbf{I})$ is a normally-distributed random variable. Langevin Monte Carlo algorithms use this discrete Markov chain, which is equivalent to a gradient ascent procedure with injected Gaussian noise: the gradient drift term drives the chain towards points of $\mathcal{U}$ where $\tilde{\mathcal{L}}$ has high density, whereas the injected noise prevents the chain from collapsing to just the (local) maximum.

Due to discretization error, the Markov chain of Equation (6) is not guaranteed to converge to the same stationary distribution as the continuous process. This can be corrected by using the Metropolis-Hasting rule of Equation (3) to accept or reject states of the chain. This approach, known as the *Metropolis-adjusted Langevin algorithm* (MALA), corresponds to using a Gaussian proposal distribution with mean $\bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right)$ and covariance matrix $\epsilon\mathbf{I}$:

$$\mathcal{T}_{\text{MALA}}\left(\bar{\mathbf{u}}^{(t)} \to \bar{\mathbf{v}}\right) = \mathcal{N}\left(\bar{\mathbf{v}}; \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right), \epsilon\mathbf{I}\right). \quad (7)$$

Adaptation. The Markov chain of Equation (6) implicitly assumes that the coordinates of $\bar{\mathbf{u}}$ are uncorrelated and have approximately equal variance; that is, that $\tilde{\mathcal{L}}$ is locally isotropic. This can result in slow convergence when these conditions are violated. This behavior can be ameliorated by modifying Equation (6) to use a positive-definite *preconditioning matrix* $\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)$ [Roberts and Stramer 2002]:

$$\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right) + \sqrt{\epsilon}\sqrt{\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)}\bar{\mathbf{w}}. \quad (8)$$

The matrix $\sqrt{\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)}$ can be computed as the Cholesky decomposition of $\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)$. The corresponding proposal distribution becomes:

$$\mathcal{T}_{\text{MALA}}\left(\bar{\mathbf{u}}^{(t)} \to \bar{\mathbf{v}}\right) = \mathcal{N}\left(\bar{\mathbf{v}}; \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)\nabla_{\bar{\mathbf{u}}}\tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right), \epsilon\mathbf{M}\left(\bar{\mathbf{u}}^{(t)}\right)\right). \quad (9)$$

As the notation suggests, the preconditioning matrix can vary with $\bar{\mathbf{u}}$. We focus on this case, which we refer to as *adaptive preconditioning*, and to the corresponding algorithm as *MALA with adaptation.*

The preconditioning matrix should be selected to be representative of the correlations that exist between the dimensions of the sampling domain $\mathcal{U}$. In gradient-based optimization, this matrix is ideally set equal to the inverse of the Hessian matrix $\mathbf{H} \equiv \left[\frac{\partial^2 \tilde{\mathcal{L}}}{\partial \bar{\mathbf{u}}_i \partial \bar{\mathbf{u}}_j}\right]$, as in Newton's method [Nocedal and Wright 2006]. However, the Hessian matrix cannot be used for preconditioning in MALA, as it is not guaranteed to be positive-definite. Approaches to circumvent this include using the expectation of the Hessian to form the Fisher information matrix [Girolami and Calderhead 2011], or applying transformations to the Hessian's eigenvalues to produce a positive definite matrix [Betancourt 2013; Li et al. 2015]. Unfortunately, these Hessian-based approaches introduce a significant computational overheard compared to standard LMC: first, they require computing second-order derivatives of $\tilde{\mathcal{L}}$, instead of just first-order derivatives; second, they require performing expensive matrix operations (e.g., eigendecomposition of the Hessian). We note additionally that, even when an eigendecomposition is not required, it will be necessary to compute a factorization of the preconditioning matrix to sample the Gaussian proposal of Equation (9). As all of these computations need to be performed for every sampled path, they can become prohibitively expensive for MCMC rendering.

ALGORITHM 1: MALA with adaptation

```
Input: step-size $\epsilon > 0$.
Output: $\{\bar{\mathbf{u}}^{(t)}\}_{t=1:T}$.
/* Initialization */
1  sample random $\bar{\mathbf{u}}^{(1)}$;
2  for $t = 1 : T - 1$ do
3      sample $\eta \sim \mathrm{Unif}(\cdot; [0, 1])$;
4      if $\eta < \pi_{local}$ then
5          $\tilde{\mathbf{g}}^{(t)} \leftarrow \nabla_{\bar{\mathbf{u}}} \tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right)$;   // compute gradient
           /* Compute pc. matrix $\mathbf{M}^{(t)}$ and drift vector $\mathbf{m}^{(t)}$ */
6          sample $\bar{\mathbf{w}} \sim \mathcal{N}(\cdot; \mathbf{0}, \mathbf{I})$;
7          $\bar{\mathbf{v}} \leftarrow \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon \mathbf{M}^{(t)} \odot \mathbf{m}^{(t)} + \sqrt{\epsilon}\left(\mathbf{M}^{(t)}\right)^{\circ\frac{1}{2}} \odot \bar{\mathbf{w}}$;   // gen. proposal
8      else
9          sample $\bar{\mathbf{v}} \sim \mathcal{T}_{\mathrm{global}}\left(\bar{\mathbf{u}}^{(t)} \to \cdot\right)$;
10     end
11     $\bar{\mathbf{u}}^{(t+1)} \leftarrow$ MetropolisHastings$\left(\bar{\mathbf{v}}, \bar{\mathbf{u}}^{(t)}\right)$;   // accept or reject (Equation (3))
12 end
```

Our contributions. To alleviate these performance considerations, in this paper we will introduce three variants of MALA with adaptation that satisfy the following set of properties:

P.1 They provide adaptation using a *diagonal* adaptive preconditioning matrix, as well as an *adaptive drift vector* that will not necessarily be parallel to the gradient $\nabla_{\bar{\mathbf{u}}} \tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(t)}\right)$.

P.2 Both of these adaptation mechanisms will require minimal processing of only previously-computed first-order gradients.

P.3 The resulting MALA processes will remain ergodic, with the same stationary distribution as standard MALA.

The combination of properties **P.1** and **P.2** will ensure that our algorithms are efficient enough to facilitate MCMC rendering, avoiding second-order differentiation and expensive matrix factorizations. As we will show, this can be achieved without compromising on the quality of the sampled paths. Our three algorithms will take the general form of Algorithm 1, with the computation of the preconditioning matrix and drift vector taking place in an online fashion (Section 5), a cache-driven fashion, or a hybrid of the two (Section 6). We note that Algorithm 1 incorporates the large-step mutation of Equation (4), for which we will follow prior work (though see also Section 9). Additionally, the algorithm uses a modified version of the update of Equation (8), to emphasize the computational simplicity of diagonal preconditioning: $\odot$, $\oslash$, and $\circ\frac{1}{2}$ are element-wise multiplication, division, and square root, respectively, $\mathbf{1}$ is a vector of all ones, and we overload notation to write $\mathbf{M}^{(t)}$ as a vector.

Hamiltonian Monte Carlo. We briefly discuss the relationship of LMC to another class of techniques known as Hamiltonian Monte Carlo (HMC) [Betancourt 2017; Duane et al. 1987; Neal et al. 2011]. HMC uses Hamilton's equations to evolve a continuous-time Markov process, corresponding to a dynamical system with potential energy determined by the target function $\tilde{\mathcal{L}}$, and kinetic energy determined by a custom mass matrix. In practice, the continuous evolution equations are approximated using multiple steps of the leapfrog integrator. This results in a discrete-time Markov chain, which is combined with the Metropolis-Hastings rule. When only one integration step is used between proposals, this Markov chain becomes equivalent to MALA, with the mass matrix used for preconditioning in Equation (8) [Girolami and Calderhead 2011; Neal et al. 2011]. The use of multiple steps can help reduce sample correlation, but is prohibitively expensive for MCMC rendering, as explained by Li et al. [2015]. They instead derive a single-step procedure using a Gaussian approximation to the potential energy; as we discussed above, this is equivalent to MALA with Hessian-based preconditioning.

5 ONLINE ADAPTATION

Our goal in this section is to develop *online* adaptation mechanisms for MALA that continuously update the preconditioning matrix and drift vector as new samples are generated, while also satisfying properties **P.1-P.3**. For adaptation to be beneficial, the preconditioning matrix should adapt to the local geometry of the sampling domain, and the drift vector should accelerate convergence to local modes.

For inspiration on how to achieve these desiderata, we can utilize the similarity between the MALA transition distribution and gradient-based optimization. For example, we could draw an analogy with classical *quasi-Newton* optimization algorithms, which strike a balance between the fast convergence of the Hessian-based Newton's method, and the computational efficiency of standard first-order gradient descent. Quasi-Newton algorithms achieve this by using, at each iteration, the accumulated history of first-order gradients, to approximate the Hessian matrix. Unfortunately, naively converting these algorithms to a MALA procedure (e.g., by adding noise to each iteration) will not be successful: First, most common Hessian approximations, such as the BFGS, DFP, and Broyden methods [Nocedal and Wright 2006], are not guaranteed to be positive definite; enforcing positive-definitiveness requires expensive matrix operations [Simsekli et al. 2016; Zhang and Sutton 2011], which contradicts our desired property **P.2**. Second, even if positive definitiveness could be guaranteed, the resulting Markov chain would not necessarily be ergodic, as required by our desired property **P.3**.

We develop our online adaptation procedure by drawing ideas from two distinct areas: First, we take advantage of the similarity between the MALA transition distribution and gradient-based optimization. In particular, we propose to use the Hessian approximations and momentum terms of state-of-the-art stochastic gradient descent (SGD) algorithms [Duchi et al. 2011; Kingma and Ba 2014], as the preconditioning matrix and drift vector, respectively, of MALA with adaptation. Second, we look at the theory of *controlled MCMC* [Andrieu and Thoms 2008; Roberts and Rosenthal 2009] and use it to modify the online computation of the preconditioning matrix and drift vector in a way that ensures ergodicity. We refer to the resulting algorithm as *MALA with online adaptation.* As we develop our algorithm, we use Figure 2 as a two-dimensional visualization of the effect of each of its components on sampling performance.

Adam preconditioning. Current state-of-the-art SGD algorithms use an adaptive step-size matrix that was originally introduced in the *AdaGrad* algorithm [Duchi et al. 2011], and was later modified in algorithms such as *Adam* [Kingma and Ba 2014]. Even though these algorithms are popular, e.g., for training deep learning pipelines, they have not previously been used for MCMC rendering. We will

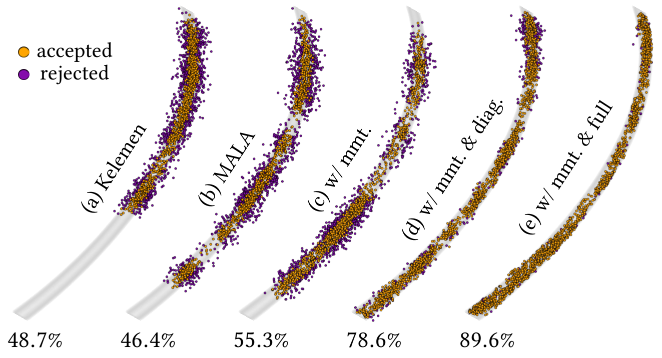

Fig. 2. **2D sampling example:** We compare various MCMC sampling algorithms on a simple two-dimensional anisotropic geometry, typical of those encountered in the primary sample space near caustics. Starting at the same initial position (top of the curve), we show 2048 samples and acceptance ratios produced by each algorithm. "Mmt." refers to momentum, and "diag." and "full" to diagonal and full preconditioning, respectively.

describe the *Adam* algorithm in the context of MALA, where the step-size matrix plays the same role as a preconditioning matrix.

Consider a set of samples $\bar{\mathbf{u}}^{(1)}, \ldots, \bar{\mathbf{u}}^{(t)}$ produced by the Markov chain. For each of the samples, we denote by $\tilde{\mathbf{g}}^{(\tau)} \equiv \nabla_{\bar{\mathbf{u}}} \tilde{\mathcal{L}}\left(\bar{\mathbf{u}}^{(\tau)}\right), \tau = 1, \ldots, t$, the gradient of the log-path contribution at that point. These gradients are already computed when generating the MALA proposals. Adam first forms a diagonal *accumulation matrix* as: [2]

$$\mathbf{G}_{\mathrm{O}}^{(t)} = \sum_{\tau=1}^{t} \beta^{\tau-1} (1-\beta) \operatorname{diag}\left(\tilde{\mathbf{g}}^{(\tau)} \cdot \left(\tilde{\mathbf{g}}^{(\tau)}\right)^{\top}\right). \tag{10}$$

The constant $\beta \in [0,1]$ results in accumulation with exponentially-decaying weights, to emphasize more recent derivatives (likely to be closer to the current location $\bar{\mathbf{u}}^{(t)}$), and downweight older ones. In practice, the accumulation is performed iteratively, without the need to store previous gradients (see Algorithm 2). Adam uses $\mathbf{G}_{\mathrm{O}}^{(t)}$ to approximate the Hessian and preconditioning matrix at $\bar{\mathbf{u}}^{(t)}$ as:

$$\mathbf{H}_{\mathrm{O}}^{(t)} = \delta \mathbf{I} + \sqrt{\mathbf{G}_{\mathrm{O}}^{(t)}}, \tag{11}$$

$$\mathbf{M}_{\mathrm{O}}^{(t)} = \left(\mathbf{H}_{\mathrm{O}}^{(t)}\right)^{-1}, \tag{12}$$

where $\delta > 0$ is a small constant.

We make the following observations for Equations (10)-(12): First, the computation of the preconditioning matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ only uses previously computed first-order gradients. Second, $\mathbf{M}_{\mathrm{O}}^{(t)}$ is guaranteed to be positive definite, as required for MALA with adaptation, without the need for additional eigendecomposition operations. Third, both $\mathbf{M}_{\mathrm{O}}^{(t)}$ and its factorization can be computed using simple *scalar* inversion and square root operations. From these, we conclude that the preconditioning matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ satisfies properties **P.1-P.2**. Finally, we refer to the AdaGrad and Adam papers [Duchi et al. 2011; Kingma and Ba 2014] for theoretical arguments justifying the Hessian approximation of Equations (10)-(12). All of these make $\mathbf{M}_{\mathrm{O}}^{(t)}$ a suitable preconditioning matrix for MALA with adaptation.

[2]We note that Adam includes a *bias correction* term in the accumulation matrix. We omit this term, as we found empirically that it results in worse rendering performance.

Diagonal versus full preconditioning. The computational advantages of diagonal adaptation come at the cost of potentially worse adaptation: the preconditioning matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ can no longer capture correlations that may exist between the dimensions of the sampling space $\mathcal{U}$. As discussed by Li et al. [2015], accounting for these correlations can improve rendering performance for complex scenes.

To quantitatively assess the extent to which adaptation with diagonal preconditioning negatively affects sampling performance, we consider a full-variant of Adam preconditioning [Duchi et al. 2011]. This requires modifying the accumulation matrix as:

$$\mathbf{G}_{\mathrm{O}}^{(t)} = \sum_{\tau=1}^{t} \beta^{\tau-1} (1-\beta)\, \tilde{\mathbf{g}}^{(\tau)} \cdot \left(\tilde{\mathbf{g}}^{(\tau)}\right)^{\top}, \tag{13}$$

with Equations (11)-(12) remaining the same. The resulting matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ remains positive definite, and therefore can be used for preconditioning. Computing this full preconditioning matrix requires factorizing the matrix $\mathbf{G}_{\mathrm{O}}^{(t)}$, and inverting the matrix $\mathbf{H}_{\mathrm{O}}^{(t)}$. An additional factorization is required to compute $\sqrt{\mathbf{M}_{\mathrm{O}}^{(t)}}$ for sampling. We note that the computational overhead of these operations compared to the case of diagonal preconditioning can be alleviated by using efficient rank-two update algorithms [Brodlie et al. 1973; Zhang and Sutton 2011] to directly compute both $\mathbf{M}_{\mathrm{O}}^{(t)}$ and its factorization.

In Figure 3(e-f), we compare equal-sample renderings produced using diagonal and full preconditioning. We observed that full preconditioning provides a small improvement over diagonal preconditioning. This small improvement comes with a considerable computational overhead, resulting in more than double the runtime compared to diagonal preconditioning. In Section 8, we show additional experiments, including comparisons to the Hessian-based preconditioning of Li et al. [2015], supporting these observations. Therefore, we focus on the diagonal case in the rest of the paper.

Adam momentum. In addition to a Hessian approximation, Adam uses *momentum* [Polyak 1964; Sutskever et al. 2013] to accelerate optimization convergence. We adopt the same as an adaptive drift vector in MALA. Concretely, we compute the momentum vector as:

$$\mathbf{m}_{\mathrm{O}}^{(t)} = \sum_{\tau=1}^{t} \alpha^{\tau-1} (1-\alpha)\, \tilde{\mathbf{g}}^{(\tau)}. \tag{14}$$

where $\alpha \in [0,1]$ controls the exponential decay of older gradients. As in Equation (10), in practice gradients are accumulated iteratively. In the next section, we describe how we modify MALA to use this momentum vector and the preconditioning matrix of Equation (12).

5.1 Ensuring ergodicity

A naive way to combine the Adam preconditioning and momentum schemes with MALA would be to modify Equation (8) as:

$$\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon \mathbf{M}_{\mathrm{O}}^{(t)} \mathbf{m}_{\mathrm{O}}^{(t)} + \sqrt{\epsilon}\sqrt{\mathbf{M}_{\mathrm{O}}^{(t)}}\,\bar{\mathbf{w}}. \tag{15}$$

Unfortunately, using this equation to generate samples is likely to result in incorrect estimates, as the resulting stochastic process is not guaranteed to be ergodic, even when combined with Metropolis-Hastings. This is because the proposal distribution implied by Equation (15) is no longer *time-homogeneous*: Due to the use of the preconditioning matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ and momentum vector $\mathbf{m}_{\mathrm{O}}^{(t)}$, the proposal

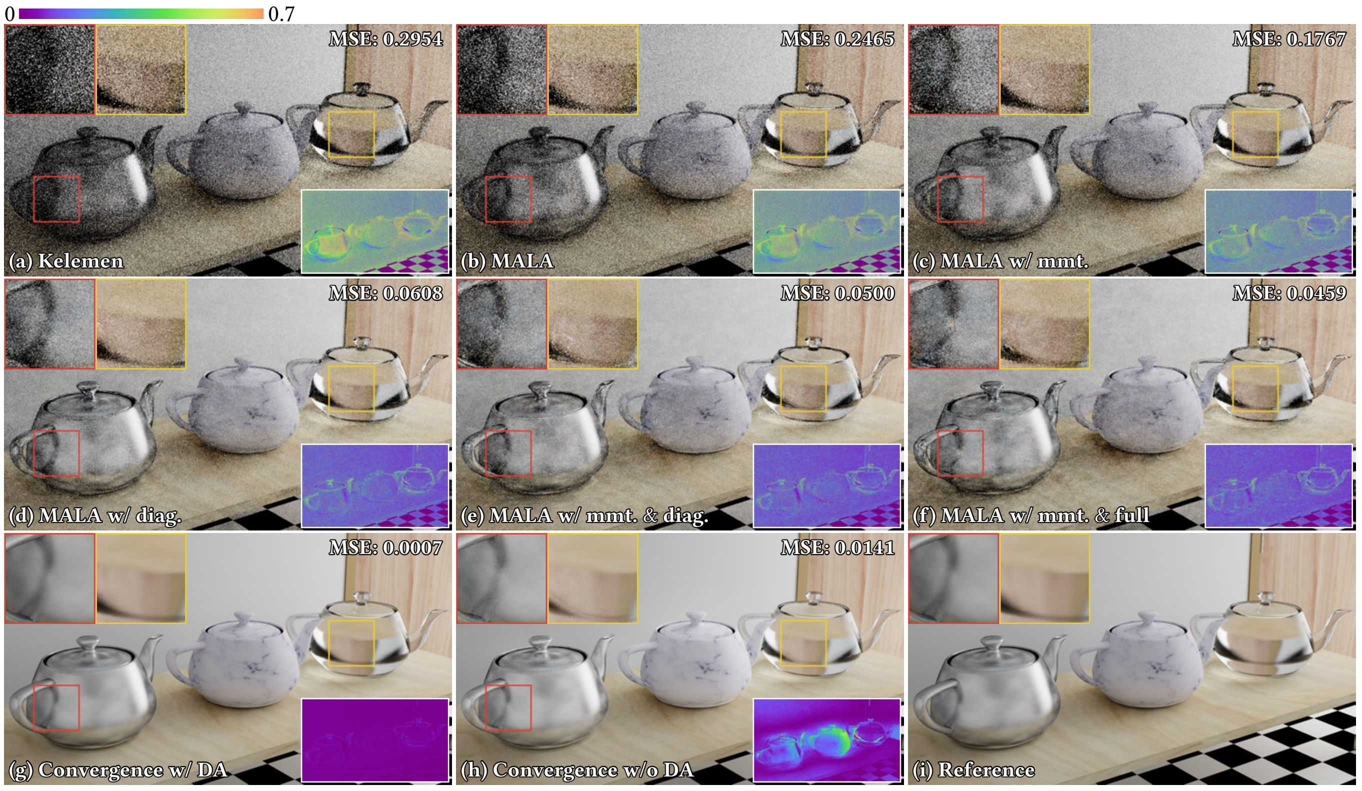

Fig. 3. **Ablation study:** We use the *door* scene to compare different versions of our proposed MALA with online adaptation. In (a-f), we show equal-sample (64 samples-per-pixel) renderings produced using the small-step perturbation of Kelemen et al. [2002], and MALA with different combinations of momentum ("mmt.") and diagonal ("diag.") or full preconditioning. In (g-h), we show converged (20, 000 samples-per-pixel) renderings produced using MALA with online adaptation, with and without diminishing adaptation ("DA"). All images include error maps with respect to a reference rendering in (i) produced using BDPT.

will be different each time the process returns to a specific state $\bar{\mathbf{u}}$. (Equivalently, the proposal is a function of not just $\bar{\mathbf{u}}$, but also the time t.) Consequently, the conditions for ergodicity we discussed in Section 3 are no longer sufficient. To address this problem, we first present some technical background on *controlled MCMC* techniques, then use this theory to derive an ergodic version of Equation (15).

Controlled MCMC. We provide a brief review, referring to Andrieu and Thoms [2008] for more details, and to Hachisuka and Jensen [2011] for a previous application of the controlled MCMC framework to rendering in the context of photon mapping.[3]

Controlled MCMC algorithms consider, in addition to the sampling domain $\mathcal{U}$, a parameter domain Θ. Parameter vectors $\theta \in \Theta$ control the Markov chain's transition distribution $\mathcal{P} : \mathcal{U}\times\mathcal{U}\times\Theta \rightarrow \mathbb{R}_{\geq}$. Each parameter vector effectively maps to a different transition distribution $\mathcal{P}\left(\bar{\mathbf{u}} \rightarrow \cdot; \theta\right)$, which can be used to sample new states for the chain. The parameter vectors to be used are determined using a time-varying *control function* $\mathcal{F}$ that maps tuples of states to parameters. At time $t+1$, given the history of samples $\bar{\mathbf{u}}^{(1)}, \ldots, \bar{\mathbf{u}}^{(t)}$ and an initial parameter vector $\theta^{(0)}$, sampling proceeds as:

$$\text{Sample } \bar{\mathbf{u}}^{(t+1)} \sim \mathcal{P}\left(\bar{\mathbf{u}} \rightarrow \cdot; \theta^{(t)}\right). \tag{16}$$

$$\text{Update } \theta^{(t+1)} = \mathcal{F}\left(t+1, \theta^{(0)}, \bar{\mathbf{u}}^{(1)}, \ldots, \bar{\mathbf{u}}^{(t+1)}\right). \tag{17}$$

[3]Controlled MCMC is often also referred to as adaptive MCMC. We do not use this term, to avoid confusion with MALA with adaptation.

In our derivation, we will use a set of three conditions that are sufficient for creating an ergodic controlled MCMC process, which we simplify and summarize as follows.

C.1 *Simultaneous ergodicity:* For all parameter vectors $\theta \in \Theta$, the transition distribution $\mathcal{P}\left(\bar{\mathbf{u}} \rightarrow \cdot; \theta\right)$ must be ergodic.

C.2 *Bounded convergence:* The space $\mathcal{U} \times \Theta$ is compact.

C.3 *Diminishing adaptation:* The time-varying transition distribution $\mathcal{P}\left(\bar{\mathbf{u}} \rightarrow \cdot; \theta^{(t)}\right)$ converges to some fixed distribution $\mathcal{P}\left(\bar{\mathbf{u}} \rightarrow \cdot; \theta^{*}\right)$ as $t \rightarrow \infty$.

These conditions were derived by Roberts and Rosenthal [2007], and we refer to their paper for their technical formulation. Whereas conditions C.1 and C.2 are typically trivially satisfied, the diminishing adaptation condition C.3 requires careful design of the controlled MCMC process. Effectively, this condition ensures that, as $t \rightarrow \infty$, the proposal distribution becomes time-homogeneous.

Enforcing diminishing adaptation. We now revisit Equation (15), which we can interpret as a controlled MCMC process: its parameter vector is the tuple $\theta^{(t)} = \left(\mathbf{m}_{\mathrm{O}}^{(t)}, \mathbf{M}_{\mathrm{O}}^{(t)}\right)$, and the control function $\mathcal{F}$ corresponds to Equations (14) and (10)-(12). Finally, the transition distribution $\mathcal{P}$ is the combination of a modified Equation (9):

$$\mathcal{T}_{\text{online}}\left(\bar{\mathbf{u}} \rightarrow \bar{\mathbf{v}}; \theta^{(t)}\right) = \mathcal{N}\left(\bar{\mathbf{v}}; \bar{\mathbf{u}} + \frac{1}{2}\epsilon \mathbf{M}_{\mathrm{O}}^{(t)}\mathbf{m}_{\mathrm{O}}^{(t)}, \epsilon \mathbf{M}_{\mathrm{O}}^{(t)}\right), \tag{18}$$

with the Metropolis-Hastings acceptance rule of Equation (3).

ALGORITHM 2: MALA with online adaptation

```
Input: exponential decay rates $\alpha$, $\beta$, diminishing adaptation exponents
       $c_1$, $c_2$, and positive constant $\delta$.
/* Initialization:                                                    */
1 begin
2     $\mathbf{G}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{m}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{d}^{(0)} \leftarrow \mathbf{0}$;
3 end
/* Compute pc. matrix $\mathbf{M}^{(t)}$ and drift vector $\mathbf{m}^{(t)}$:                */
4 begin
5     $\mathbf{G}^{(t)} \leftarrow \beta \mathbf{G}^{(t-1)} + (1-\beta)\, \tilde{\mathbf{g}}^{(t)} \odot \tilde{\mathbf{g}}^{(t)}$;
6     $\mathbf{M}^{(t)} \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\delta \mathbf{1} + t^{-c_1} \left(\mathbf{G}^{(t)}\right)^{\circ \frac{1}{2}}\right)\right)$;      // pc. matrix
7     $\mathbf{d}^{(t)} \leftarrow \alpha \mathbf{d}^{(t-1)} + (1-\alpha)\, \tilde{\mathbf{g}}^{(t)}$;
8     $\mathbf{m}^{(t)} \leftarrow t^{-c_2}\, \mathbf{d}^{(t)} + \tilde{\mathbf{g}}^{(t)}$;      // momentum
9 end
```

This controlled MCMC process satisfies the simultaneous ergodicity condition C.1: For each θ, the proposal distribution of Equation (18) is equivalent to MALA with a fixed preconditioning matrix and mean shift that, when combined with Metropolis-Hastings, is ergodic. It also satisfies the bounded convergence condition C.2, because $\mathcal{U}$ is bounded and closed, and Θ is in practice finite (floating point numbers have finite state space [Hachisuka and Jensen 2011]).

However, this controlled MCMC process does not satisfy the diminishing adaptation condition C.3: As $t \to \infty$, $\mathbf{M}_{\mathrm{O}}^{(t)}$ and $\mathbf{m}_{\mathrm{O}}^{(t)}$ do not converge to a limit that is either fixed or a function of only $\bar{\mathbf{u}}^{(t)}$. To address this, we propose to modify Equations (11) and (14) as:

$$\mathbf{H}_{\mathrm{O}}^{(t)} = \delta \mathbf{I} + \frac{1}{t^{c_1}} \sqrt{\mathbf{G}_{\mathrm{O}}^{(t)}}, \tag{19}$$

$$\mathbf{m}_{\mathrm{O}}^{(t)} = \tilde{\mathbf{g}}^{(t)} + \frac{1}{t^{c_2}} \sum_{\tau=1}^{t} \alpha^{\tau-1} (1-\alpha)\, \tilde{\mathbf{g}}^{(\tau)}, \tag{20}$$

where $c_1, c_2 > 0$ are positive exponents. With this modification, as $t \to \infty$, $\mathbf{M}_{\mathrm{O}}^{(t)}$ converges to $\delta \mathbf{I}$, and $\mathbf{m}_{\mathrm{O}}^{(t)}$ converges to the gradient $\tilde{\mathbf{g}}^{(t)}$. Therefore, the modified controlled MCMC process satisfies the diminishing adaptation condition C.3, and is thus ergodic. We confirm this and demonstrate the need for diminishing adaptation in Figure 3(g-i): when rendering converged images with and without diminishing adaptation, the former matches the reference (rendered with BDPT), whereas the latter results in strong artifacts.

We term this modified controlled MCMC process *MALA with online adaptation*. We summarize the online adaptation procedure in Algorithm 2, using the same notational conventions as Algorithm 1.

Discussion. In Figures 3(a-e), we show equal-sample comparisons of different versions of our algorithm against two baselines, the small-step perturbation of Kelemen et al. [Kelemen et al. 2002], and standard MALA without adaptation. We observe that both of the adaptation mechanisms we introduced, adaptive drift and (especially) preconditioning, result in significantly improved rendering performance. Their combined use in MALA with online adaptation reduces error by almost 80% compared to standard MALA. This improvement comes at a negligible computational overhead—the runtimes for Figures 3(b) and (e) differ by approximately 8%. Additionally, in Section 8, we perform equal-time comparisons of our algorithm with state-of-the-art MCMC rendering algorithms (including a version of MALA with Hessian-based preconditioning [Li et al. 2015]). Despite its simplicity, our algorithm results in $2-4\times$ error reduction across a variety of complex scenes.

At the same time, we can identify two shortcomings of MALA with online adaptation. First, the diminishing adaptation scheme of Equations (19)-(20) has the effect that, as $t \to \infty$, our algorithm converges to standard MALA without preconditioning or momentum (Equation (7)). This can be problematic for complicated scenes where rendering low-noise images requires a very large number of samples, as later samples will not benefit from preconditioning.

Second, as discussed in Section 3 and shown in Algorithm 1, in practice it is necessary to combine the MALA-based small-step perturbation with a large-step mutation. When a large step is performed and the Markov chain moves to a new neighborhood in $\mathcal{U}$, the accumulated preconditioning matrix $\mathbf{M}_{\mathrm{O}}^{(t)}$ and momentum vector $\mathbf{m}_{\mathrm{O}}^{(t)}$ will not be good approximations of the local geometry until a few more small steps are performed within this neighborhood. As a result, we expect that our adaptive preconditioning will not be effective immediately after the occurrence of large steps, resulting in many rejected samples. Even though we can mitigate this effect by reducing the probability π_{global} of a large step, this would result in slower global exploraion. In the next section, we build upon the techniques we developed in this section to propose a version of MALA with adaptation that overcomes both of the shortcomings discussed above, without hindering global exploration.

6 CACHE-DRIVEN AND HYBRID ADAPTATION

In this section, we show how to modify MALA with online adaptation to achieve better asymptotic performance as $t \to \infty$ and reduce disruption by large steps, while still producing an algorithm that satisfies properties P.1-P.3. Our main insight is as follows: introducing diminishing adaptation in the previous section was necessary because Equations (10) and (14) accumulate gradients using exponentially-decaying weights. This accumulation scheme serves as a computationally efficient proxy for using only past gradients that were computed in the proximity of the current sample. If we can afford to store and query all previously-computed gradients, then we can modify these rules to only use nearby previous gradients, without the need for exponentially-decaying accumulation.

We formalize this insight by making two contributions: First, we combine a new gradient caching scheme with the adaptation rules of the previous section to propose a new algorithm that we call *MALA with cache-driven adaptation*, and which remains ergodic without the need for diminishing adaptation. Second, we develop a hybrid between online and cache-driven adaptation, that combines their complementary advantages. The resulting hybrid algorithm will be the main algorithm we use in our rendering experiments.

Gradient caching. At any given time t, we denote by

$$C^{(t)} \equiv \left\{ \left(\bar{\mathbf{u}}^{(\tau)}, \mathbf{g}^{(\tau)} \right), \tau = 1, \ldots, t \right\}, \tag{21}$$

the set of tuples of sequences $\bar{\mathbf{u}}^{(\tau)}$ and corresponding gradient values $\mathbf{g}^{(\tau)}$ that the Markov chain has visited. We refer to this set as a *gradient cache*, hinting at its eventual implementation in Section 7.

Given a sequence $\bar{\mathbf{v}}$ and a radius $r > 0$, we denote by

$$Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right) \equiv \left\{\tau : \left(\bar{\mathbf{u}}^{(\tau)}, \mathbf{g}^{(\tau)}\right) \in C^{(t)} \text{ and } \left\|\bar{\mathbf{u}}^{(\tau)} - \bar{\mathbf{v}}\right\| < r\right\}, \tag{22}$$

the set of integer indices for tuples in $C^{(t)}$ whose sequence $\bar{\mathbf{u}}^{(\tau)}$ is within Euclidean distance r of $\bar{\mathbf{v}}$. Continuing the cache analogy, we refer to this set as the *query* of cache $C^{(t)}$ for point $\bar{\mathbf{v}}$.

Cache-driven preconditioning and momentum. We can now update the definitions of preconditioning and momentum in Section 5 so that they use gradient caching. In particular, we define diagonal preconditioning with gradient caching as:

$$\mathbf{G}_{\mathrm{C}}^{(t)} = \operatorname{diag}\left(\frac{1}{\left|Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)\right|} \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)} \cdot \left(\tilde{\mathbf{g}}^{(\tau)}\right)^{\top}\right), \tag{23}$$

$$\mathbf{H}_{\mathrm{C}}^{(t)} = \delta \mathbf{I} + \sqrt{\mathbf{G}_{\mathrm{C}}^{(t)}}, \tag{24}$$

$$\mathbf{M}_{\mathrm{C}}^{(t)} = \left(\mathbf{H}_{\mathrm{C}}^{(t)}\right)^{-1}, \tag{25}$$

where $\left|Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)\right|$ is the size of the set $Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)$. Similarly, we define momentum with gradient caching as:

$$\mathbf{m}_{\mathrm{C}}^{(t)} = \frac{1}{\left|Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)\right|} \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)}. \tag{26}$$

Finally, with these definitions at hand, we modify the MALA update rule of Equation (8) to take the form:

$$\bar{\mathbf{u}}^{(t+1)} = \bar{\mathbf{u}}^{(t)} + \frac{1}{2}\epsilon \mathbf{M}_{\mathrm{C}}^{(t)} \mathbf{m}_{\mathrm{C}}^{(t)} + \sqrt{\epsilon}\sqrt{\mathbf{M}_{\mathrm{C}}^{(t)}}\bar{\mathbf{w}}, \tag{27}$$

Equivalently, given a cache $C^{(t)}$, we can define a controlled variant of the MALA proposal distribution of Equation (9) as:

$$\mathcal{T}_{\text{cache}}\left(\bar{\mathbf{u}} \to \bar{\mathbf{v}}; C^{(t)}\right) = \mathcal{N}\left(\bar{\mathbf{v}}; \bar{\mathbf{u}} + \frac{1}{2}\epsilon \mathbf{M}_{\mathrm{C}}^{(t)} \mathbf{m}_{\mathrm{C}}^{(t)}, \epsilon \mathbf{M}_{\mathrm{C}}^{(t)}\right). \tag{28}$$

The combination of this proposal with the Metropolis-Hastings acceptance rule is our proposed MALA with cache-driven adaptation.

Ergodicity. Analogously to Section 5.1, MALA with cache-driven adaptation is a controlled MCMC procedure: the parameter vector is the gradient cache, $\theta^{(t)} \equiv C^{(t)}$, and the control function $\mathcal{F}$ corresponds to Equations (23)-(26). In Section 5.1, we had to modify the control equations for online preconditioning and momentum, to ensure that the diminishing adaptation condition **C.3** is satisfied. By contrast, this is not necessary for the cache-driven variants we defined above, as ergodicity can be achieved in an alternative way.

In particular, we can define some threshold H on the maximum number of entries $\left|C^{(t)}\right|$ we will allow the cache to store. Once this threshold is reached, we stop caching, while continuing to run MALA with the preconditioning matrix and momentum of Equations (23)-(26). Consequently, the time-varying transition distribution of Equation (28) reaches a fixed distribution when the threshold H is reached, and therefore satisfies the diminishing adaptation condition **C.3**. We conclude that MALA with cache-driven adaptation is ergodic with the correct stationary distribution. This mechanism for ensuring ergodicity is known as *finite adaptation* [Roberts and Rosenthal 2007], and remains effective for any finite value of H. We summarize the cache-driven adaptation procedure in Algorithm 3, using the same notational conventions as Algorithm 1.

ALGORITHM 3: MALA with cache-driven adaptation

```
Input: radius $r$, and positive constant $\delta$.
/* Initialization: */
begin
    $C^{(0)} \leftarrow \varnothing$;
end

/* Compute pc. matrix $\mathbf{M}^{(t)}$ and drift vector $\mathbf{m}^{(t)}$: */
begin
    $C^{(t)} \leftarrow C^{(t-1)}$;
    if $\left|C^{(t)}\right| < H$ then
        $C^{(t)} \leftarrow C^{(t)} \cup \left\{\left(\bar{\mathbf{u}}^{(t)}, \tilde{\mathbf{g}}^{(t)}\right)\right\}$;   // update gradient cache
    end
    $s \leftarrow \left|Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)\right|^{-1}$;
    $\mathbf{G}^{(t)} \leftarrow s \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)} \odot \tilde{\mathbf{g}}^{(\tau)}$;
    $\mathbf{M}^{(t)} \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\delta \mathbf{1} + \left(\mathbf{G}^{(t)}\right)^{\circ \frac{1}{2}}\right)\right)$;   // pc. matrix
    $\mathbf{m}^{(t)} \leftarrow s \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)}$;   // momentum
end
```

Comparing online and cache-driven adaptation. We have now defined two versions of MALA with adaptation, one in Section 5 using an online procedure inspired from SGD, and one in this section driven by a gradient cache. The cache-driven version overcomes the two limitations of the online version discussed at the end of Section 5: in particular, given that it does not require introducing explicit diminishing adaptation, the cache-driven version converges to MALA with a non-unit preconditioning matrix—approximately the one that would result from an MCMC process using the online scheme without large-step mutations or exponentially-decaying summation. Additionally, the cache-driven version adapts to the occurrence of large-step mutations seamlessly, as Equations (23)-(26) allow it to immediately switch to computing preconditioning matrices and momentum vectors using any available gradients in the new neighborhood it moves to. We show a two-dimensional visualization of this improved behavior in Figure 4.

Conversely, these advantages come at the cost of maintaining the gradient cache $C^{(t)}$, and performing potentially expensive cache queries $Q\left(\cdot; C^{(t)}, r\right)$ for each sampled path. This cost increases as rendering progresses and the cache becomes larger, making the addition of new entries and queries more expensive. Another downside of the cache-driven scheme is that at the initial stages of the rendering process, when the cache is sparsely populated, it may result in no preconditioning even in cases where the online scheme would produce a non-identity preconditioning matrix. This can happen, for example, after a sequence of small-step perturbations that are all greater than the radius r. These often occur immediately after a large step mutation, when the Markov chain lands at a new neighborhood and starts quickly converging towards the local mode: Whereas the online scheme would use the resulting sequence of gradients to form a preconditioning matrix, the cache-driven scheme would reject them when quering for nearby gradients.

Finally, the effectiveness of cache-driven adaptation will decrease as the path length B, and thus dimensionality $r(B)$ of the cached gradients, increases. This is because of the *curse of dimensionality*,

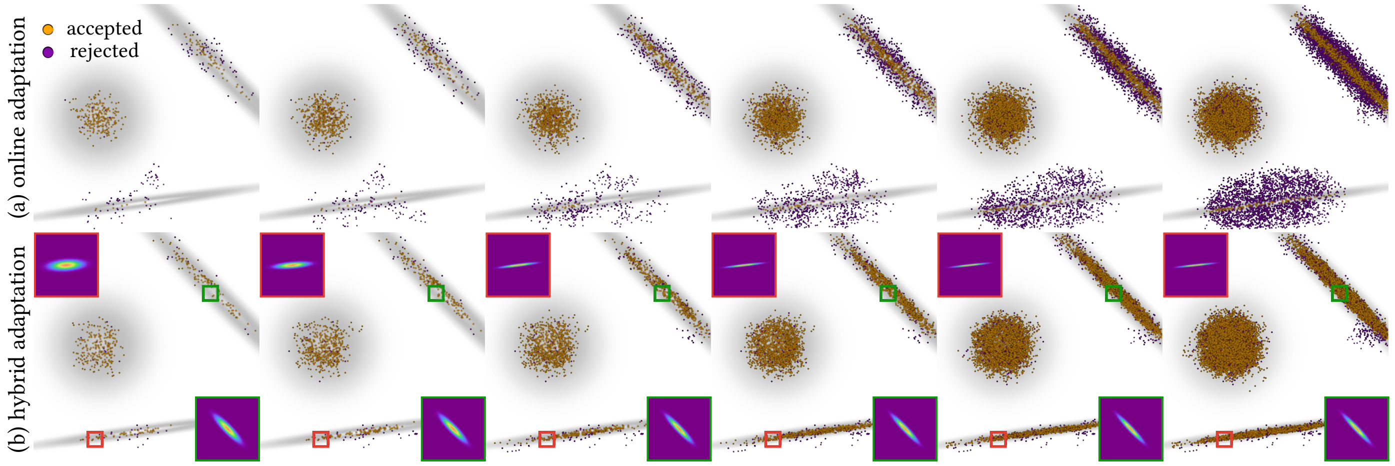

Fig. 4. **2D Gaussian mixture model:** We compare the performance of MALA with online and hybrid adaptation, on a two-dimensional disconnected geometry, typical of the isolated "islands" of primary sample space. Both algorithms use full preconditioning and large-step mutation probability $\mathcal{T}_{\text{global}} = 0.1$. Each row shows sample distributions when running the algorithm for (from left to right) 0.5K, 1K, 2K, 4K, 8K and 16K samples. In the bottom row, the insets show the evolution of the anisotropic transition kernels at two different points in space. Please zoom in to better compare the results.

which states that the number of samples that need to be cached for nearest-neighbor estimation to be accurate increases exponentially with dimensionality. As we discuss in Section 7, in practice we circumvent this issue by disabling adaptation for very long paths, as is commonly done in many MCMC rendering techniques [Müller et al. 2019; Reibold et al. 2019; Zheng and Zwicker 2019].

Hybrid adaptation. We propose a hybrid between the two adaptation schemes that combines their complementary advantages. Our hybrid operates in two stages, as shown in Algorithm 4.

The first stage continues while the number of entries in the cache, $\left|C^{(t)}\right|$, is smaller than some predefined threshold H. During this stage, we use online adaptation as in Algorithm 2, but without diminishing adaptation ($c_1, c_2 = 0$), as ergodicity will be guaranteed when we switch to the second stage. Additionally, we continue to expand the cache with any new gradient evaluation.

The second stage starts once the number of entries in the case reaches the threshold, $\left|C^{(t)}\right| \geq H$. During this stage, we disable online adaptation and use exclusively the cache-driven version. We additionally stop adding new entries to the cache, which ensures ergodicity and helps reduce the cost of subsequent cache queries.

ALGORITHM 4: MALA with hybrid adaptation

```
Input: exponential decay rates $\alpha$, $\beta$, radius $r$, positive constant $\delta$.
/* Initialization: */
begin
    $\mathbf{G}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{m}^{(0)} \leftarrow \mathbf{0}$, $C^{(0)} \leftarrow \varnothing$;
end

/* Compute pc. matrix $\mathbf{M}^{(t)}$ and drift vector $\mathbf{m}^{(t)}$: */
begin
    $C^{(t)} \leftarrow C^{(t-1)}$;
    if $\left|C^{(t)}\right| < H$ then
        /* online */
        $C^{(t)} \leftarrow C^{(t)} \cup \left\{\left(\bar{\mathbf{u}}^{(t)}, \tilde{\mathbf{g}}^{(t)}\right)\right\}$;   // update gradient cache
        $\mathbf{G}^{(t)} \leftarrow \beta \mathbf{G}^{(t-1)} + (1-\beta)\, \tilde{\mathbf{g}}^{(t)} \odot \tilde{\mathbf{g}}^{(t)}$;
        $\mathbf{M}^{(t)} \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\delta \mathbf{1} + \left(\mathbf{G}^{(t)}\right)^{\circ \frac{1}{2}}\right)\right)$;   // pc. matrix
        $\mathbf{m}^{(t)} \leftarrow \alpha \mathbf{m}^{(t-1)} + (1-\alpha)\, \tilde{\mathbf{g}}^{(t)}$;   // momentum
    else
        /* cache-driven */
        $s \leftarrow \left|Q\left(\bar{\mathbf{v}}; C^{(t)}, r\right)\right|^{-1}$;
        $\mathbf{G}^{(t)} \leftarrow s \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)} \odot \tilde{\mathbf{g}}^{(\tau)}$;
        $\mathbf{M}^{(t)} \leftarrow \operatorname{diag}\left(\mathbf{1} \oslash \left(\delta \mathbf{1} + \left(\mathbf{G}^{(t)}\right)^{\circ \frac{1}{2}}\right)\right)$;   // pc. matrix
        $\mathbf{m}^{(t)} \leftarrow s \sum_{\tau \in Q\left(\bar{\mathbf{u}}^{(t)}; C^{(t)}, r\right)} \tilde{\mathbf{g}}^{(\tau)}$;   // momentum
    end
end
```

7 IMPLEMENTATION DETAILS

We implement our proposed algorithms within the open-source reference implementation [Li 2015] of H2MC [Li et al. 2015]. This is a primary-sample-space MCMC renderer that, critically, uses automatic differentiation [Bell 2015] to compute derivatives of the measurement contribution function. We modify the automatic differentiation code to disable the computation of Hessian matrices, which are needed by H2MC but not by our algorithms. Our implementation is available at the project website [Luan et al. 2020].

Perturbation and mutation strategies. We use the same large-step mutation and small-step perturbation strategies as H2MC, including their proposed parameterization change and mixing weights. We only apply MALA-based sampling to the small-step mutation, as suggested by H2MC. We also follow H2MC and disable MALA with adaptation for paths of length $B \geq 8$ (primary-sample-space sequences of length $r(B) \geq 16$), reverting to Gaussian mutations [Kelemen et al. 2002] for longer paths. We note that we did not implement lens perturbations in our algorithm, which we also disabled when comparing with H2MC. This lets us perform fair comparisons with other primary-sample-space MLT methods [Bitterli et al. 2018; Hachisuka et al. 2014] that do not support lens perturbations.

Gradient caching. We implement the gradient cache with the nanoflann library [Blanco and Rai 2014], which uses KD-tree structures to support efficient nearest-neighbor searches even in high-dimensional spaces (such as the primary sample space). To further reduce the overhead of cache queries, we use a hybrid of nearest-neighbor and radius queries: We terminate the KD-tree traversal once there are K neighbors found within the search radius r.

Truncated gradients. Livingstone and Zanella [2019] recently analyzed the convergence behavior of gradient-based MCMC (e.g., Langevin and Hamiltonian Monte Carlo), and showed that very large variations in the magnitude of the gradients can significantly deteriorate convergence. This observation is also supported by older literature, which suggests truncating gradients when using either standard MALA [Roberts and Tweedie 1996] or MALA with adaptation [Atchade 2006]. We follow this suggestion and truncate all dimensions of our gradients to a maximum value of $g = 100$, which helps ensure robust convergence across a large variety of scenes.

Parameter settings. Algorithms 1-4 use a few parameters, namely: exponential-decay rates (α, β), diminishing adaptation exponents (c_1, c_2), cache query radius (r), nearest neighbors (K), cache size (H), uniform preconditioning weight (δ), and step size (ϵ). We set $\alpha = 0.9$, $\beta = 0.999$, and $\delta = 0.001$, which are the values recommended by Kingma and Ba [2014]. We also select values $\epsilon = 0.01$, $r = 0.01$, $K = 30$, and $H = 10,000$ for the remaining parameters. We use these parameter values for all the renderings shown in this paper.

8 EXPERIMENTS

We run all experiments on a machine with an Intel Xeon E5-2630 v3 processor. We render all reference images by running BDPT for more than 12 hours per scene. **The supplemental document shows additional renderings, and statistics. The supplemental HTML viewer shows the full set of rendering results, and includes an interactive viewer to facilitate comparisons.** The viewer is based on the supplement of Bitterli et al. [2018]. Both supplements are available at the project website [Luan et al. 2020].

Equal-time comparisons. We perform equal-time comparisons of the online (Algorithm 2) and hybrid (Algorithm 4) versions of MALA with adaptation (Algorithm 1), with BDPT and five state-of-the-art MCMC rendering algorithms: the path-space MEMLT algorithm [Jakob and Marschner 2012], and the primary-sample-space RJMLT [Bitterli et al. 2018], MMLT [Hachisuka et al. 2014], and H2MC [Li et al. 2015] algorithms. We use a set of 17 scenes with complex illumination, occlusions, caustics, and interreflections. These characteristics make these scenes challenging for non-MCMC rendering algorithms such as BDPT. To verify the consistency of all the MCMC algorithms we test, we confirmed that in all scenes they match the reference when run until convergence. On *all* scenes, MALA with hybrid adaptation was the best performing algorithm, and MALA with online adaptation performed second best. Relative performance among the other MCMC algorithms varied from scene to scene. These results suggest that, despite its simplicity, MALA with online adaptation can be expected to have robust and efficient performance across a large variety of scenes. Additional performance gains can be obtained by using MALA with hybrid adaptation, which however has higher implementation complexity. Overall, our two algorithms resulted in an average MSE improvement of 3× compared to the third best. Finally, BDPT had the worst average performance, suggesting that the complexity of our scenes makes it necessary to use MCMC rendering for good performance. Figures 1 and 5 show a representative sample of these renderings for all the MCMC algorithms; we provide the complete set of comparisons, including comparisons to BDPT, in the supplement.

Motion blur simulations. We follow Li et al. [2015] and simulate motion blur by treating time as another dimension in the primary sample space. We then perform an equal-time comparison of our algorithms with H2MC, on two scenes that include challenging motion blur effects, as shown in Figure 6 (we provide full-sized renderings and comparisons in the supplement): the *cars* scene contains a static and a moving car, and the *necklace* scene shows a gemstone rotating and a golden ring moving linearly. In addition to motion blur, both scenes include other difficult light transport effects, such as specular-diffuse-specular (SDS) paths and complex caustics. Our algorithms significantly outperformed H2MC in both scenes, resulting in 7.9× and 7.4× MSE improvements. This demonstrates the ability of our algorithms to adapt to each dimension of the primary sample space, including time. We note that we did not compare against the other MCMC rendering algorithms we considered in this section: RJMLT does not provide an implementation that supports motion blur, and Li et al. [2015] demonstrated that H2MC outperforms MMLT and MEMLT in scenes with motion blur effects.

Evaluation of preconditioning schemes. As we discussed in Section 5, our algorithms use a diagonal preconditioning matrix to increase the computational efficiency of generating new samples, at the cost of potentially worse adaptation compared to full preconditioning. To better characterize this tradeoff between sampling efficiency and adaptation quality, we use a set of nine scenes to perform additional experiments comparing our diagonal and full preconditioning, as well as the Hessian-based preconditioning of Li et al. [2015]. First, we perform equal-sample and equal-time comparisons of the three preconditioning schemes combined with our MALA with online adaptation. The equal-sample comparisons allow us to evaluate the quality of generated samples, without concern for computational complexity; whereas the equal-time comparisons allow us to evaluate overall performance, taking into account both sample quality and computational cost. In the equal-sample comparisons, Hessian-based and full preconditioning resulted in an average MSE improvement of 1.3× and 1.1× compared to diagonal preconditioning, suggesting that both schemes produce better samples than diagonal preconditioning. However, in the equal-time comparisons, Hessian-based and full preconditioning resulted in 2.4× and 2.1× worse average MSE than diagonal preconditioning. Therefore, the improvement in sample quality that Hessian-based and full preconditioning provide is outweighed by their increased computational complexity. These observations hold consistently across all scenes, supporting our use of diagonal preconditioning. As discussed in Section 5, for full preconditioning, the increased computational complexity is due to the matrix operations (factorization and inversion) required to generate proposals. For the Hessian-based preconditioning, this increased complexity is further compounded by second-order differentiation. Full preconditioning provides a mid-point

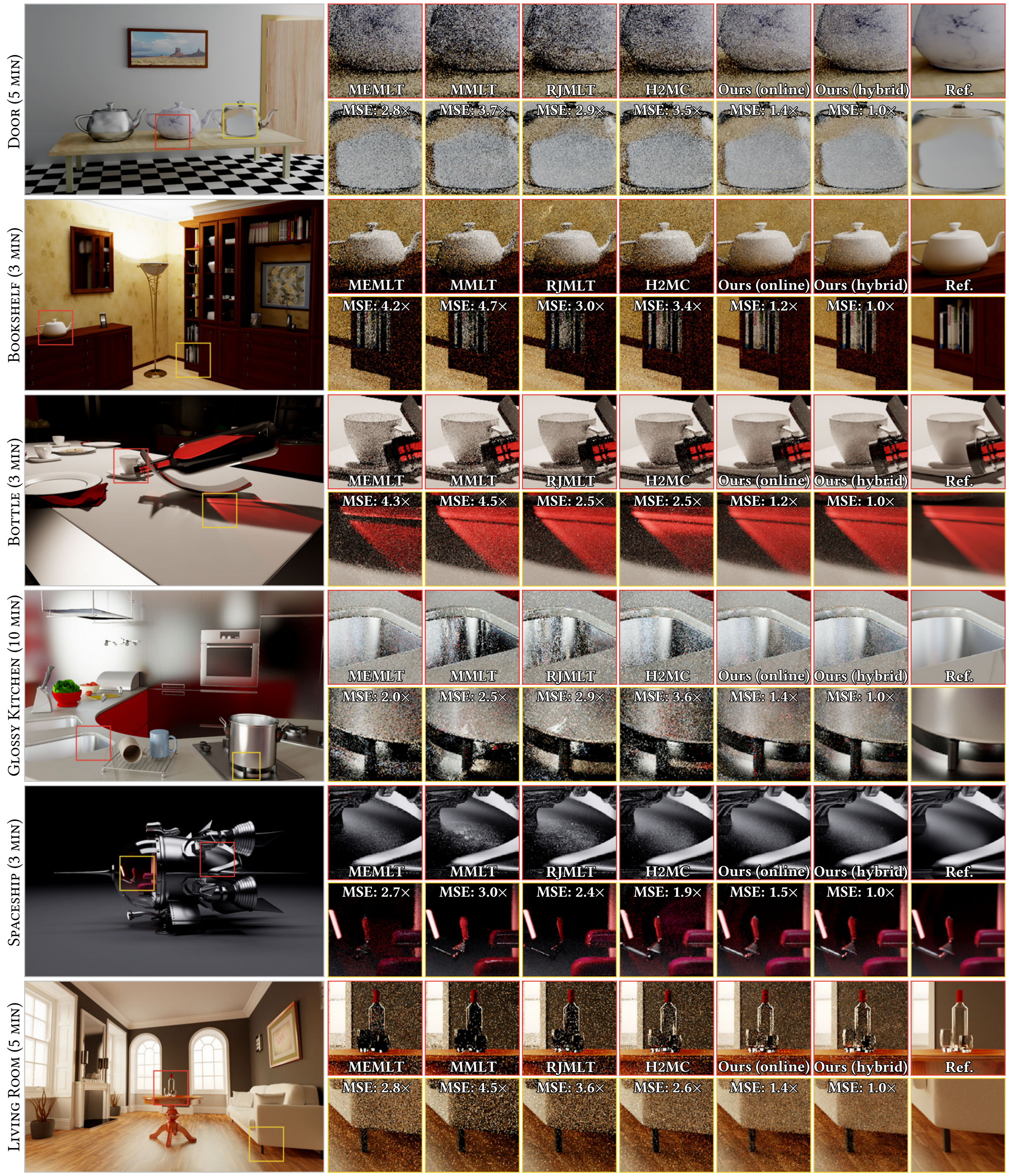

Fig. 5. **Equal-time comparisons:** We compare MEMLT [Jakob and Marschner 2012], MMLT [Hachisuka et al. 2014], RJMLT [Bitterli et al. 2018], H2MC [Li et al. 2015] and two of our algorithms, across several scenes with complex illumination and occlusion, glossy caustics and interreflections.

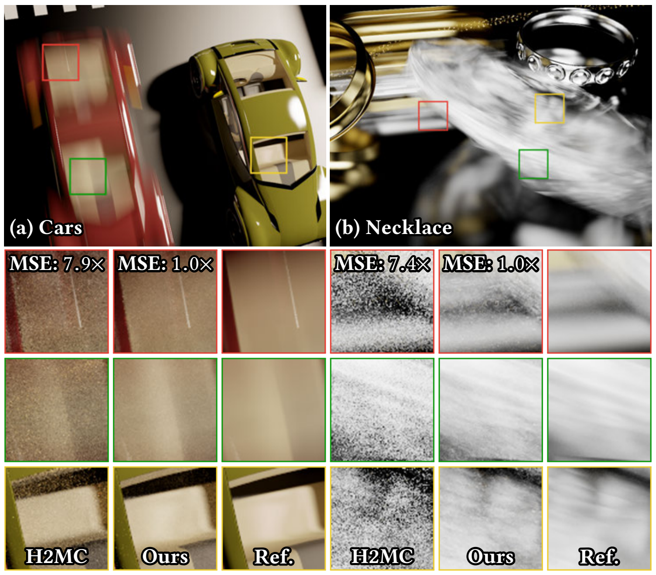

Fig. 6. **Motion blur effects:** Equal-time comparison of H2MC and our hybrid adaptation on the *cars* (3 minutes) and *necklace* (7 minutes) scenes.

between the high sample quality of Hessian-based preconditioning, and the computational efficiency of diagonal preconditioning. Figure 7 compares renderings for a representative scene; we provide the complete set of comparisons in the supplement.

Second, we compare equal-sample and equal-time renderings produced using our proposed MALA with hybrid adaptation, combined with either our diagonal or Hessian-based preconditioning. In the latter case, during the second stage of our hybrid algorithm, we estimate the Hessian by averaging the Hessian matrices cached at nearby points. Therefore, caching completely removes the need for second-order differentiation during the second stage; we note, though, that sample generation still requires expensive matrix operations (enforcing positive-definitiveness, factorization, and inversion). Compared to our diagonal preconditioning, Hessian-based preconditioning resulted in 1.6× better average MSE in equal-sample renderings, and 2.5× worse average MSE in equal-time renderings. We observe that, even though the use of caching helps reduce the computational overhead of Hessian-based preconditioning during the latter stage of the rendering process, this overhead is still significant enough to outweigh any performance gains from the improved sample quality. Therefore, these experiments suggest that using diagonal preconditioning is preferable even for cache-driven adaptation. Figure 8 compares renderings for one representative scene, and the complete set of comparisons is available in the supplement.

9 LIMITATIONS AND FUTURE WORK

We discuss some important aspects of our algorithms, which suggest limitations and directions for future exploration.

Differentiation in rendering. Compared to a standard primary-sample-space MCMC renderer, the main additional requirement of our adaptation algorithms is the ability to compute first-order gradients of path tracing algorithms. This differentiation functionality is becoming standard in modern rendering engines [Anderson et al. 2017; Che et al. 2018; Li et al. 2018, 2015; Nimier-David et al. 2019; Zhang et al. 2019], which incorporate automatic differentiation [Griewank and Walther 2008] to facilitate both forward and inverse rendering. There is additionally active research [Jakob 2019] on designing automatic differentiation libraries that are better suited to the computational graphs typical of rendering, compared to general-purpose libraries such as the one in our implementation [Bell 2015]. These developments suggest that our online adaptation algorithm can be readily incorporated within modern rendering software, and potentially offer even bigger performance improvements over gradient-free algorithms than those reported in Section 8.

Gradient-based sampling in path space. Our algorithms are currently designed for use only in the primary sample space. This is in large part due to the mathematical convenience this space provides, making it amenable to differentiation. Considering the complementary advantages of sampling techniques operating in the two spaces, it would be interesting to explore the use of MALA with adaptation in path space, where gradients have already found some use for sampling near-specular paths [Jakob and Marschner 2012; Kaplanyan et al. 2014]. As discussed above, this exploration can benefit from the availability of end-to-end differentiable rendering systems.

Global exploration. We focused on developing small-step perturbation proposals for local exploration, and relied on prior work [Kelemen et al. 2002] for the large-step mutation proposals needed for global exploration. As both critically affect rendering performance, in the future it will be important to research global exploration techniques that synergize with the local exploration techniques we developed. One direction is to use the cache of our hybrid method to additionally facilitate global exploration. A straightforward way to do this would be to modify the cache to store both gradients and throughput values $f(\bar{\mathbf{u}})$. We could use this information to replace the uniform large-step mutation proposal $\mathcal{T}_{\text{global}}$ with one that uses the cache for importance sampling, for example, by forming a kernel density estimate of $f(\bar{\mathbf{u}})$. This form of *cache-driven adaptation for global exploration* can potentially improve rendering performance, at a negligible overhead (the cache is already used for local exploration) and without affecting ergodicity (the same arguments for the ergodicity of cache-driven local exploration apply). We show a proof-of-concept demonstration in Figure 9, where all images are rendered with the same settings as in Figure 3. We leave a more detailed investigation, including utilizing learning-based techniques for path guiding in the primary sample space [Müller et al. 2019; Reibold et al. 2019; Zheng and Zwicker 2019], for the future.

Parameter setting. As discussed in Section 7, our algorithms requires setting a large number of parameters. Empirically, we observed that some of these parameters (for example, the step size ϵ) can result in significant deterioration in performance if set incorrectly. However, we additionally found that selecting parameter values can be done once, without the need for per-scene fine-tuning: We coarsely searched for parameter values that produced good performance on a simple scene (*ring* scene in the supplement), and then used the same values for all of our experiments. This suggests that users of our implementation should be able to achieve robust high performance, without the need to change the default parameter values we provide. In the supplement, we empirically

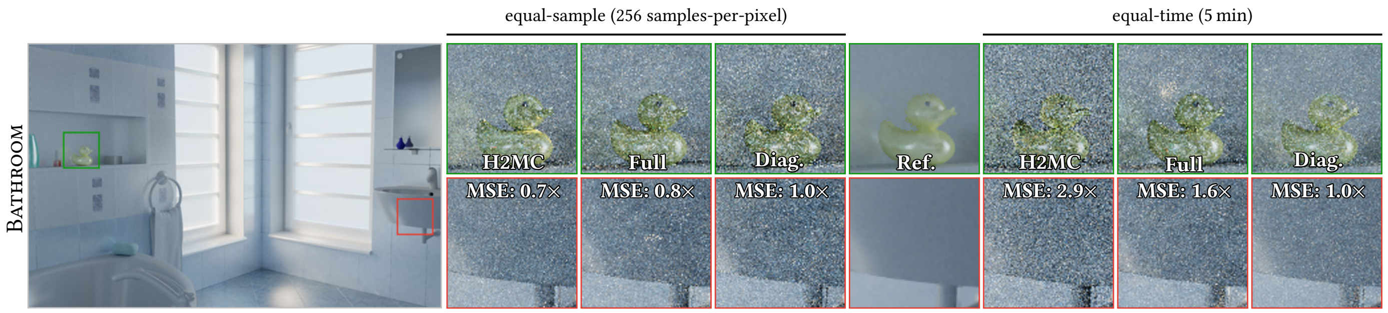

Fig. 7. **Evaluation of preconditioning schemes for online adaptation:** We use the *bathroom* scene for equal-sample and equal-time comparisons of diagonal, full, and H2MC preconditioning, combined with online adaptation.

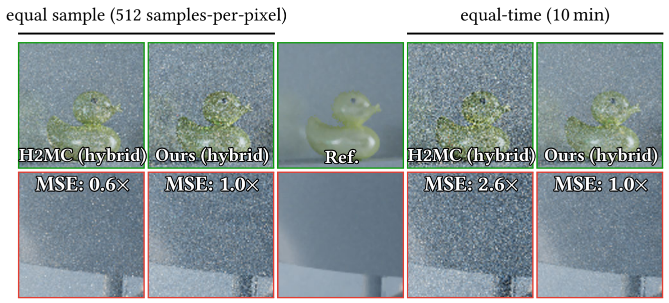

Fig. 8. **Evaluation of preconditioning schemes for hybrid adaptation:** We use the *bathroom* scene for equal-sample and equal-time comparisons of diagonal and H2MC preconditioning, combined with hybrid adaptation. The scene and insets are the same as those in Fig. 7.

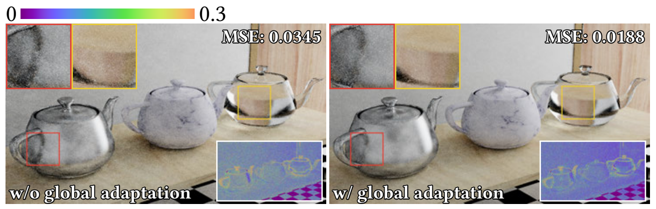

Fig. 9. **Global exploration:** Equal-sample (64 samples-per-pixel) renderings of the *door* scene, comparing MALA with hybrid adaptation, without (left) and combined with (right) cache-driven global exploration (Section 9).

investigate how changing different parameters impacts rendering performance. An interesting future research direction would be to use controlled MCMC for online adaptation (during the rendering process) of parameters towards their scene-specific optimal values.

10 CONCLUSION

We introduced two variants of the Metropolis-adjusted Langevin algorithm with adaptation that are suitable for MCMC rendering. For efficiency, our algorithms mimic SGD and employ diagonal preconditioning and momentum, computed using only past gradients in an online or cache-driven manner. To ensure correctness, our algorithms are designed using the framework of controlled MCMC. Put together, these features allow our algorithms to adapt to the geometry of the primary sample space for local exploration, and to seamlessly combine with complementary sampling techniques for global exploration. We have demonstrated that our algorithms outperform the state-of-the-art in equal-time comparisons, resulting in an average MSE improvement of 3× on a variety of scenes, and 7× on scenes with motion blur. We hope that, as differentiation support becomes commonplace in rendering systems, our paper and publicly-available implementation [Luan et al. 2020] will motivate further research on using gradient-based MCMC for rendering.

ACKNOWLEDGMENTS

This work was supported by NSF grants 1730147, 1900783, 1900849, 1900927, and a gift from AWS Cloud Credits for Research.

REFERENCES

Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. 2017. Aether: An Embedded Domain Specific Sampling Language for Monte Carlo Rendering. *ACM TOG* (July 2017).
Christophe Andrieu and Johannes Thoms. 2008. A tutorial on adaptive MCMC. *Statistics and computing* (2008).
James Arvo. 1986. Backward ray tracing. *ACM SIGGRAPH Course Notes* (1986).
James Arvo. 1994. The irradiance Jacobian for partially occluded polyhedral sources. *SIGGRAPH* (1994).
Yves F Atchade. 2006. An adaptive version for the Metropolis adjusted Langevin algorithm with a truncated drift. *MCAP* (2006).
Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. 2019. Inverse Path Tracing for Joint Material and Lighting Estimation. *CVPR* (2019).
Bradley M Bell. 2003-2015. Cppad: A package for differentiation of C++ algorithms.
Michael Betancourt. 2013. A general metric for Riemannian manifold Hamiltonian Monte Carlo. *International Conference on Geometric Science of Information* (2013).
Michael Betancourt. 2017. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434* (2017).
Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. 2018. Reversible jump metropolis light transport using inverse mappings. *ACM TOG* (2018).
Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively metropolised Monte Carlo light transport simulation. *ACM TOG* (2019).
Jose Luis Blanco and Pranjal Kumar Rai. 2014. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees.
Ken W Brodlie, AR Gourlay, and John Greenstadt. 1973. Rank-one and rank-two corrections to positive definite matrices expressed in product form. *IMA Journal of Applied Mathematics* (1973).
Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of markov chain monte carlo.* CRC press.
Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2018. Inverse Transport Networks. *arXiv preprint arXiv:1809.10820* (2018).
Changyou Chen, David Carlson, Zhe Gan, Chunyuan Li, and Lawrence Carin. 2016. Bridging the gap between stochastic gradient MCMC and stochastic optimization. *Artificial Intelligence and Statistics* (2016).
Min Chen and James Arvo. 2000. Theory and application of specular path perturbation. *ACM TOG* (2000).
Tianqi Chen, Emily Fox, and Carlos Guestrin. 2014. Stochastic gradient hamiltonian monte carlo. *ICML* (2014).
Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. 1987. Hybrid monte carlo. *Physics letters B* (1987).
John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* (2011).
Alain Durmus, Umut Simsekli, Eric Moulines, Roland Badeau, and Gaël Richard. 2016. Stochastic gradient Richardson-Romberg Markov chain Monte Carlo. *NeurIPS* (2016).
Philip Dutre, Philippe Bekaert, and Kavita Bala. 2006. *Advanced global illumination.* AK Peters/CRC Press.
Philip Dutré, Eric P. Lafortune, and Yves D. Willems. 1993. Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. *CGVT* (1993).

Mark Girolami and Ben Calderhead. 2011. Riemann manifold langevin and hamiltonian monte carlo methods. *JRSS* (2011).

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. *ECCV* (2016).

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse volume rendering with material dictionaries. *ACM TOG* (2013).

Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Siam.

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic progressive photon mapping. *ACM TOG* (2009).

Toshiya Hachisuka and Henrik Wann Jensen. 2011. Robust adaptive photon tracing using photon path visibility. *ACM TOG* (2011).

Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed metropolis light transport. *ACM TOG* (2014).

Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. 2015. Improved half vector space light transport. In *CGF*.

W Keith Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. (1970).

Nicolas Holzschuch and François X Sillion. 1995. Accurate computation of the radiosity gradient with constant and linear emitters. *EGSR* (1995).

Homan Igehy. 1999. Tracing ray differentials. *SIGGRAPH* (1999).

Wenzel Jakob. 2019. Enoki: structured vectorization and differentiation on modern processor architectures. https://github.com/mitsuba-renderer/enoki.

Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM TOG* (2012).

Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. 2008a. Radiance caching for participating media. *ACM TOG* (2008).

Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. 2011. Progressive Photon Beams. *ACM TOG* (2011).

Wojciech Jarosz, Volker Schönefeld, Leif Kobbelt, and Henrik Wann Jensen. 2012. Theory, Analysis and Applications of 2D Global Illumination. *ACM TOG* (2012).

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. 2008b. Irradiance Gradients in the Presence of Participating Media and Occlusions. *EGSR* (2008).

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. *Rendering Techniques* (1995).

Henrik Wann Jensen. 2001. *Realistic image synthesis using photon mapping.* AK Peters/CRC Press.

James T Kajiya. 1986. The rendering equation. *SIGGRAPH* (1986).

Anton S Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM TOG* (2014).

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. *CGF* (2002).

Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2015. Matching Real Fabrics with Micro-Appearance Models. *ACM TOG* (2015).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Jaroslav Krivánek, Kadi Bouatouch, Sumanta N Pattanaik, and Jiri Zara. 2006. Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping. *Rendering Techniques* (2006).

Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance caching for efficient global illumination computation. *IEEE TVCG* (2005).

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying Points, Beams, and Paths in Volumetric Light Transport Simulation. *ACM TOG* (2014).

Eric P. Lafortune and Yves D. Willems. 1993. Bi-directional path tracing. *Compugraphics* (1993).

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain metropolis light transport. *ACM TOG* (2013).

Don Lemons and Anthony Gythiel. 1997. Paul Langevin's 1908 paper "On the Theory of Brownian Motion". *American Journal of Physics* (1997).

Tzu-Mao Li. 2015. dpt. https://github.com/BachiLi/dpt/.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM TOG* (2018).

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics. *ACM TOG* (2015).

Samuel Livingstone and Giacomo Zanella. 2019. On the robustness of gradient-based MCMC algorithms. *arXiv preprint arXiv:1908.11812* (2019).

Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Project Website. https://research.cs.cornell.edu/langevin-mcmc.

Yi-An Ma, Tianqi Chen, and Emily Fox. 2015. A complete recipe for stochastic gradient MCMC. *NeurIPS* (2015).

Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. 2018. Second-Order Occlusion-Aware Volumetric Radiance Caching. *ACM TOG* (2018).

Gisiro Maruyama. 1955. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo* (1955).

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* (1953).

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. *CGF* (2017).

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM TOG* (2019).

Radford M Neal et al. 2011. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo* (2011).

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: a retargetable forward and inverse renderer. *ACM TOG* (2019).

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization.* Springer.

Bernt Øksendal. 2003. Stochastic differential equations. In *Stochastic differential equations.* Springer.

Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. 2018. Geometry-Aware Metropolis Light Transport. *ACM TOG* (2018).

Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. 2017. Fusing State Spaces for Markov Chain Monte Carlo Rendering. *ACM TOG* (2017).

Jacopo Pantaleoni. 2017. Charted Metropolis Light Transport. *ACM TOG* (2017).

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation.* Morgan Kaufmann.

Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *U. S. S. R. Comput. Math. and Math. Phys.* (1964).

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A First-Order Analysis of Lighting, Shading, and Shadows. *ACM TOG* (2007).

Florian Reibold, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2019. Selective Guided Sampling with Complete Light Transport Paths. *ACM TOG* (2019).

Gareth O Roberts and Jeffrey S Rosenthal. 2007. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability* (2007).

Gareth O Roberts and Jeffrey S Rosenthal. 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* (2009).

Gareth O Roberts and Osnat Stramer. 2002. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability* (2002).

Gareth O Roberts and Richard L Tweedie. 1996. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* (1996).

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-Based Error Control for Irradiance Caching. *ACM TOG* (2012).

Martin Šik, Hisanari Otsu, Toshiya Hachisuka, and Jaroslav Křivánek. 2016. Robust light transport simulation via metropolised bidirectional estimators. *ACM TOG* (2016).

Umut Simsekli, Roland Badeau, Taylan Cemgil, and Gaël Richard. 2016. Stochastic quasi-newton langevin monte carlo. *ICML* (2016).

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. *ICML* (2013).

Frank Suykens and Yves D Willems. 2001. Path differentials and applications. *Rendering Techniques* (2001).

Chia-Yin Tsai, Aswin C Sankaranarayanan, and Ioannis Gkioulekas. 2019. Beyond Volumetric Albedo–A Surface Optimization Framework for Non-Line-Of-Sight Imaging. *CVPR* (2019).

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

Eric Veach and Leonidas Guibas. 1995. Bidirectional estimators for light transport. *Photorealistic Rendering Techniques* (1995).

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. *SIGGRAPH* (1997).

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-Line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM TOG* (2014).

Gregory J Ward and Paul S Heckbert. 1992. *Irradiance gradients.* Technical Report.

Gregory J Ward, Francis M Rubinstein, and Robert D Clear. 1988. A ray tracing solution for diffuse interreflection. *SIGGRAPH* (1988).

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. *ICML* (2011).

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM TOG* (2019).

Yichuan Zhang and Charles A. Sutton. 2011. Quasi-Newton Methods for Markov Chain Monte Carlo. *NeurIPS* (2011).

Shuang Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. 2016. Downsampling scattering parameters for rendering anisotropic media. *ACM TOG* (2016).

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. *CGF* (2019).