Anisotropic Gaussian mutations for metropolis light transport through Hessian-Hamiltonian dynamics

Anisotropic Gaussian Mutations for Metropolis Light Transport through Hessian-Hamiltonian Dynamics

Tzu-Mao Li
MIT CSAIL

Jaakko Lehtinen
Aalto University
NVIDIA

Ravi Ramamoorthi
University of California, San Diego

Wenzel Jakob
ETH Zurich

Frédo Durand
MIT CSAIL

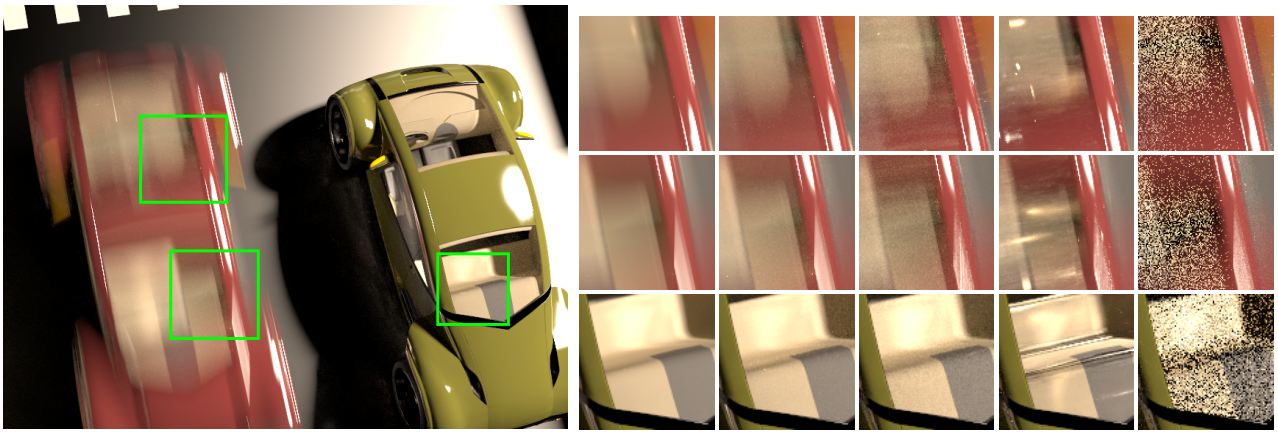


Figure 1: CARS*: Equal-time (20 minutes) comparison on the cars scene, with a static car and a moving car lit by an area light. The direct lighting is computed separately. The interior of the car is enclosed by near-specular glass windows, which gives rise to specular-diffuse-specular (SDS) paths that are challenging to sample. The three insets show the renderings of our method (H^2MC), Manifold Exploration Metropolis Light Transport (MEMLT), Multiplexed Metropolis Light Transport (MMLT), and Bidirectional Path Tracing (BDPT). The reference (REF) is rendered by our method in roughly 15 hours. BDPT cannot efficiently sample the sparse contribution function and suffers from severe noise. MMLT tends to get trapped in the hard-to-find features and produces correlated noise. MEMLT specializes in finding difficult static specular paths, but does not consider the anisotropy in the time domain, resulting in ghosting artifacts. Our method can efficiently resolve the hard-to-find SDS paths like the specialized method, and is more general so that it can resolve moving caustic paths in the window by capturing the correlation between the time domain and path space.*

Abstract

The simulation of light transport in the presence of multi-bounce glossy effects and motion is challenging because the integrand is high dimensional and areas of high-contribution tend to be narrow and hard to sample. We present a Markov Chain Monte Carlo (MCMC) rendering algorithm that extends Metropolis Light Transport by automatically and explicitly adapting to the local shape of the integrand, thereby increasing the acceptance rate. Our algorithm characterizes the local behavior of throughput in path space using its gradient as well as its Hessian. In particular, the Hessian is able to capture the strong anisotropy of the integrand. We obtain the derivatives using automatic differentiation, which makes our solution general and easy to extend to additional sampling dimensions such as time.

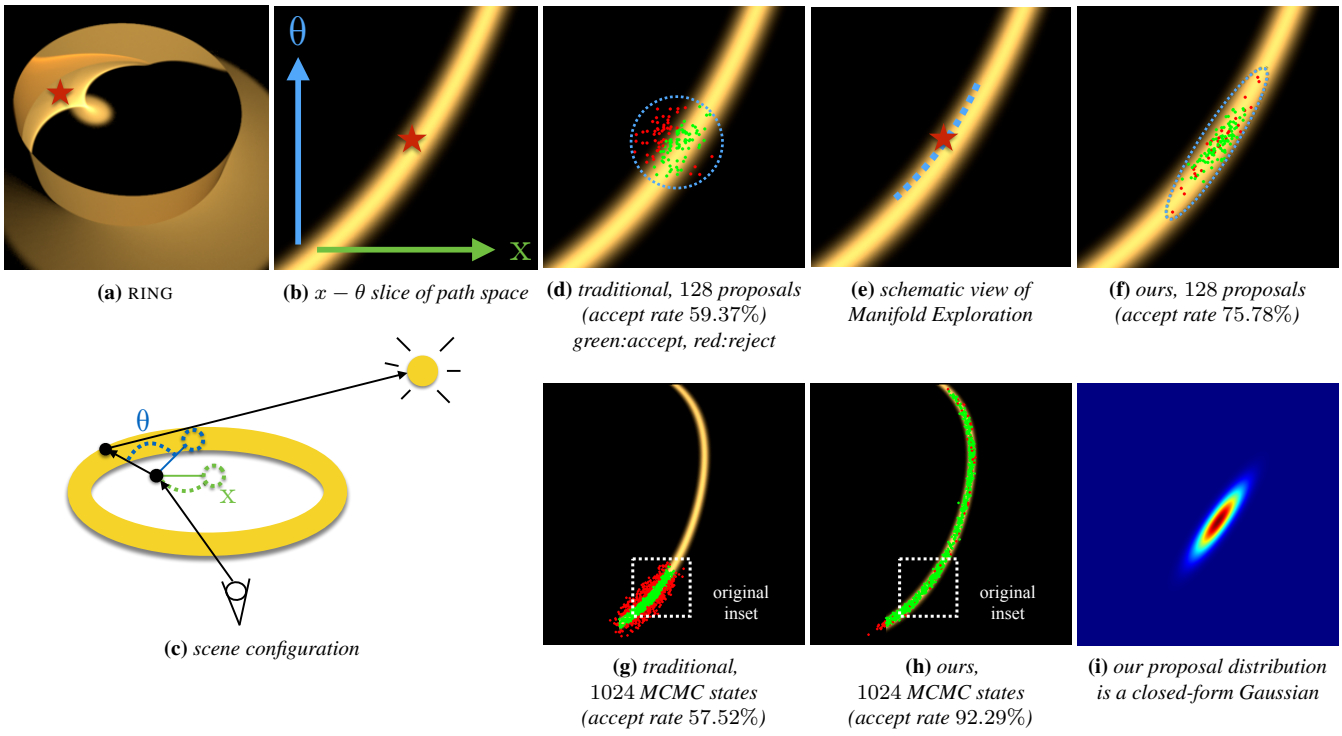
However, the resulting second order Taylor expansion is not a proper distribution and cannot be used directly for importance sampling. Instead, we use ideas from Hamiltonian Monte-Carlo and simulate the Hamiltonian dynamics in a flipped version of the Taylor expansion where gravity pulls particles towards the high-contribution region. Whereas such methods usually require numerical integration, we show that our quadratic landscape leads to a closed-form anisotropic Gaussian distribution for the final particle positions, and it results in a standard Metropolis-Hastings algorithm. Our method excels at rendering glossy-to-glossy reflections on small and highly curved surfaces. Furthermore, unlike previous work that derives sampling anisotropy with pen and paper and only considers specific effects such as specular BSDFs, we characterize the local shape of throughput through automatic differentiation. This makes our approach very general. In particular, our method is the first MCMC rendering algorithm that is able to resolve the anisotropy in the time dimension and render difficult moving caustics.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Ray Tracing

Keywords: global illumination, Metropolis light transport

1 Introduction

Light transport phenomena such as caustics, multiple-bounce glossy transport and motion blur often concentrate high contributions in a narrow volume within the high-dimensional sample space. While efficient methods exist for local importance sampling of individual scattering events, their combined effect on path throughput is intricate and hard to sample, leading to noisy images. Figure 2 shows a caustic caused by a glossy gold ring. The integrand (Figure 2 (b)) is sparse: for points on the floor (x), only a few incident directions



(a) RING **(b)** $x - \theta$ *slice of path space* **(d)** *traditional,* 128 *proposals (accept rate* 59.37%*) green:accept, red:reject* **(e)** *schematic view of Manifold Exploration* **(f)** *ours,* 128 *proposals (accept rate* 75.78%*)*

(c) *scene configuration*

(g) *traditional,* 1024 *MCMC states (accept rate* 57.52%*)* **(h)** *ours,* 1024 *MCMC states (accept rate* 92.29%*)* **(i)** *our proposal distribution is a closed-form Gaussian*

Figure 2: RING EXAMPLE: ***(a)*** *A motivating example showing the caustics caused by a highly-glossy gold ring, lit by a distant point light.* ***(b)*** *A slice of the two-bounce indirect light field around the red star, where x represents one of the dimensions in screen-space, and θ represents one of the dimensions along the BRDF sampling direction (the configuration is shown in* ***(c)****). The path contribution is sparse, and most of the contributions are zero.* ***(d)*** *The green/red dots represent the accepted/rejected proposal samples of a traditional MCMC rendering algorithm [Kelemen et al. 2002], which uses isotropic mutation that makes the sampling inefficient.* ***(e)*** *We also show the schematic of Manifold Exploration [Jakob and Marschner 2012], which only travels on the tangent of a lower dimensional space.* ***(f)*** *Our approach builds a Gaussian approximation around the neighborhood, enabling us to efficiently traverse the target function locally. Some samples are rejected due to the adaptivity, but it still results in a higher acceptance rate.* ***(g)(h)*** *We show the zoomed out slice (the positions of the original insets* ***(d)*** *and* ***(e)*** *are at the bottom of the images) with dots now representing the samples obtained by simulating the Markov chain for* 1024 *states; our method explores the space more thoroughly. We also show the false color visualization of the Gaussian approximation in* ***(i)****, which takes the width of the function into consideration.*

(θ) contribute radiance through reflection. Even in this simple scene, sparsity makes standard numerical integration methods inefficient. The region of high-contribution is continuous, but highly anisotropic, and the anisotropy varies over the integrand. In this paper, we present a general solution by extending Metropolis Light Transport [Veach and Guibas 1997] (a Markov Chain Monte Carlo sampler) to exploit the local structure of the path contribution function over its entire high-dimensional domain.

Adapting to the local anisotropic behavior of the integrand has been a long-standing challenge in rendering. Previous work has focused on model-based characterizations of anisotropy that are tied to specific effects (specular transfer, motion, etc.) [Jakob and Marschner 2012; Belcour et al. 2013; Kaplanyan et al. 2014], and combining them is not easy. Closest to our work is Manifold Exploration [Jakob and Marschner 2012] and Half-vector Space Light Transport [Kaplanyan et al. 2014; Hanika et al. 2015] which use assumptions about the mirror direction and specular reflection to derive major directions of anisotropy (Figure 2(e)), and walk along a lower-dimensional manifold. In contrast, we seek a general solution that can characterize the "thickness" of the manifold in all directions, avoiding case-specific manual derivations.

Adapting to the local behavior of the integrand boils down to two main problems: 1) characterizing the anisotropy using local information and 2) sampling according to the derived information. We solve 1) by characterizing the local throughput using its derivatives. Since the gradient provides weak directional information, we also use the second derivative, the Hessian matrix. Whereas the gradient points only into the direction of the strongest increase, the Hessian additionally captures the correlation between coordinates. While the Hessian has been used before in rendering, e.g. [Holzschuch and Sillion 1998; Schwarzhaupt et al. 2012], its manual derivation is tedious and has usually been restricted to specific transport phenomena such as diffuse-only. In contrast, we use automatic differentiation, e.g. [Griewank and Walther 2008], which allows us to handle general effects.

While the Hessian captures anisotropy well, the second problem, sampling, remains: it is not possible to directly sample from the resulting quadratic approximation because it does not define a proper distribution and grows to infinity. Instead, we start from Hamiltonian Monte Carlo (HMC) [Duane et al. 1987], a MCMC sampling algorithm that proposes new sample locations by simulating the dynamics of a particle that starts at the current sample with a random initial velocity. The particle evolves under gravity in a landscape composed of the contribution function flipped upside down so that the particle is attracted to high contribution areas (low height) by gravity. Crucially, we do not apply HMC directly: this would be too expensive, because it would require numerical integration to generate just a single sample, and each integration time step would involve costly ray tracing, shading, and derivatives which do not

directly contribute to the image. In practice, up to a hundred time steps per sample may be needed [Neal 2010]. Instead, we apply a modified version of HMC that results in closed-form integration. As we show in the paper, running Hamiltonian dynamics on a 2nd-order function with a Gaussian distribution of initial momentums leads to a Gaussian distribution of final positions, and it results in a standard Metropolis-Hastings sampling. While traditional Metropolis sampling also uses a Gaussian distribution of proposals, it is usually isotropic and is centered on the current sample. In contrast, our Gaussian proposal is anisotropic, conforms to the shape of the contribution function, and is centered towards higher values according to the local gradient and Hessian.

Our method is general both theoretically and practically thanks to the use of the 2nd-order Taylor expansion and automatic differentiation. In particular, it can be easily extended to time for motion blur effects, so that we are able to resolve the correlation between path-space and time for a light path that contains a moving caustic in a window (Figure 1). We focus on surface rendering in this paper, though our general approach could be extended to handle a variety of other phenomena such as BSSRDFs or participating media.

In summary, the contributions of this paper are:

- A MCMC rendering algorithm that utilizes second-order derivatives, where the derivatives are obtained by automatic differentiation.
- A novel Hessian-Hamiltonian Monte Carlo rendering algorithm that combines the local second-order approximation with analytical Hamiltonian Monte Carlo and a prior Gaussian. Specifically, we show that a simulation of HMC leads to a simple anisotropic Gaussian distribution for sampling.
- In addition to the sampling method, we also propose a modified parameterization of the path space based on the *primary sample space* proposed by Kelemen *et al.* [2002]. The modified parameterization reduces the correlation between the dimensions (Figure 7).

2 Related Work

Our work is closely related to the rendering algorithms that build upon MCMC sampling and the methods that utilize derivatives to drive the sampling process.

Metropolis Light Transport. In light transport simulation, we need to compute the path integral [Veach 1998] I_j for each pixel j:

$$I_j = \int_\Omega h_j(\boldsymbol{x}) f(\boldsymbol{x}) d\mu(\boldsymbol{x}), \tag{1}$$

where Ω is *path space*, which contains all the light paths, h_j is the camera response function for pixel j, $f(\boldsymbol{x})$ is the path contribution function [Veach 1998], and $\mu(\boldsymbol{x})$ is the area density of path $\boldsymbol{x}$.

Veach and Guibas [1997] apply the Markov Chain Monte Carlo (MCMC) sampling method [Metropolis et al. 1953] by generating a sequence of MCMC samples $\boldsymbol{x}_i$. A new proposal sample is *mutated* from the previous sample, and probabilistically accepted or rejected. Specifically, given a sample $\boldsymbol{x}_{i-1}$, and a target function $f^*(\boldsymbol{x})$, which is commonly set to the luminance of $f(\boldsymbol{x})$, we first generate a *proposal* sample $\boldsymbol{x}'$ with the transition probability $Q(\boldsymbol{x}_{i-1} \to \boldsymbol{x}')$, and set the next sample $\boldsymbol{x}_i$ as follows:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{x}' & \text{with probability } a(\boldsymbol{x}_{i-1} \to \boldsymbol{x}') \\ \boldsymbol{x}_{i-1} & \text{otherwise,} \end{cases} \tag{2}$$

where the acceptance probability a is defined as

$$a(\boldsymbol{x}_{i-1} \to \boldsymbol{x}') = \min\left(1, \frac{f^*(\boldsymbol{x}') Q(\boldsymbol{x}' \to \boldsymbol{x}_{i-1})}{f^*(\boldsymbol{x}_{i-1}) Q(\boldsymbol{x}_{i-1} \to \boldsymbol{x}')}\right) \tag{3}$$

This is called the Metropolis-Hastings update rule [Hastings 1970], and it satisfies the *detailed balance* condition. That is, for any light paths $\boldsymbol{x}$ and $\boldsymbol{y}$, we have

$$f^*(\boldsymbol{x}) Q(\boldsymbol{x} \to \boldsymbol{y}) a(\boldsymbol{x} \to \boldsymbol{y}) = f^*(\boldsymbol{y}) Q(\boldsymbol{y} \to \boldsymbol{x}) a(\boldsymbol{y} \to \boldsymbol{x}). \tag{4}$$

If a transition function satisfies the detailed balance condition, and if there is a strict positive probability to sample all light paths with non-zero contribution (*the ergodicity*), it will converge to a distribution proportional to the target function $f^*(\boldsymbol{x})$. Veach and Guibas then approximated the path integral I_j at pixel j using the weighted average of the MCMC samples:

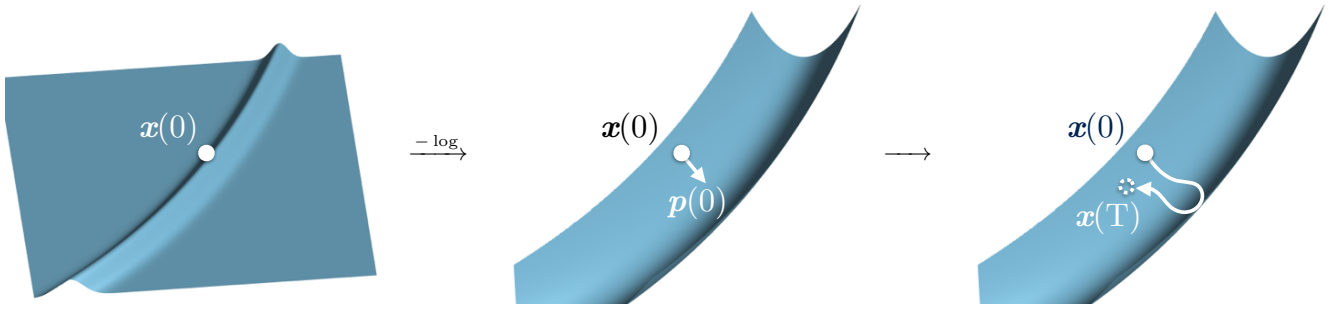
$$I_j = \frac{b}{N} \sum_{i=1}^{N} \frac{h_j(\boldsymbol{x}_i) f(\boldsymbol{x})}{f^*(\boldsymbol{x})}, \tag{5}$$

where b is a normalization constant, which is the average of $f^*(\boldsymbol{x})$ over the image. Originally Veach and Guibas designed several specialized mutation strategies to cope with different lighting scenarios. Each strategy has a different asymmetric probability distribution, which introduces a significant challenge to implement all the strategies correctly. To simplify the algorithm, Kelemen *et al.* [2002] proposed to mutate the state in the *random number space*, which makes the mutation agnostic to the particular visual effect. Unfortunately, both the mutation strategies proposed by Veach and Guibas and Kelemen *et al.* do not respect the complex local structure in the sampling domain, which makes them inefficient in some difficult cases.

Metropolis Light Transport has been extended in several aspects. Cline *et al.* [2005] proposed the Energy Redistribution Path Tracing technique by running many short Markov chains. Lai *et al.* [2007] adapted mutations with different parameters using Population Monte Carlo. Kitaoka *et al.* [2009] introduced Replica Exchange that exchanges states between multiple Markov chains to avoid getting stuck at local modes. All these methods require some form of local mutation strategies. We introduce a new local sampling strategy that adapts to the local structure of the function. Lai *et al.* [2009] proposed a temporal mutation strategy for MLT based on object-space transformation. Unlike their method, which requires a specially designed mutation, we treat the time dimension the same as the other dimensions, and handle the correlation between coordinates using the Hessian.

Jakob and Marschner [2012], Kaplanyan *et al.* [2014], and Hanika *et al.* [2015] use the first derivatives of the half-vectors of a specular light path to guide the MCMC sampling. These methods apply a form of Newton-iteration to sample new light paths satisfying certain constraints. While they improve the sampling efficiency of glossy and specular surfaces significantly, their methods can sometimes be inefficient on small, highly-curved surfaces, because of their first-order approximation. In addition, they only account for a subset of terms in the path contribution function, ignoring important effects such as the Fresnel reflection or light source emission profiles. In contrast, we utilize second-order derivatives and do not assume any particular effect. For example, we are able to render difficult moving caustics (Figure 1), where their methods would suffer from ghosting artifacts.

Hachisuka *et al.* [2014] proposed Multiplexed Metropolis Light Transport which combines MCMC methods with Multiple Importance Sampling [1995]. Their method is orthogonal to our algorithm, and we build our bidirectional path tracer based on their approach.

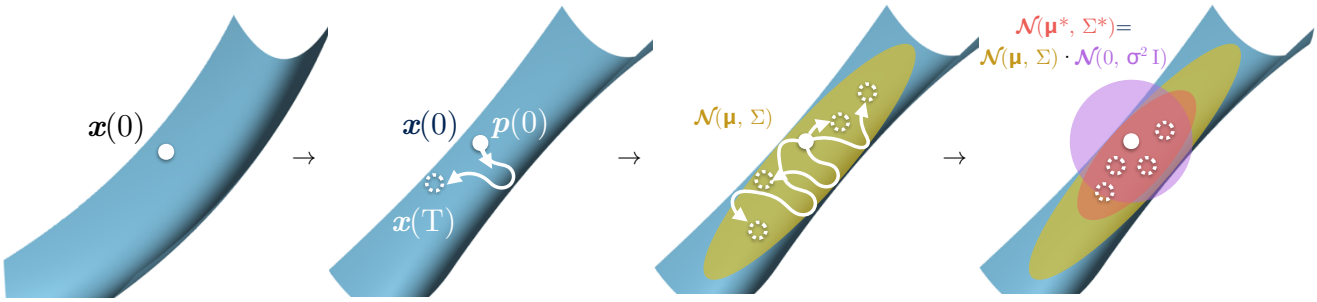


(a) *given current sample* $\boldsymbol{x}\,(0)$ *and target function*

(b) *flip function so that gravity pulls the particle, sample initial momentum* $\boldsymbol{p}\,(0)$

(c) *simulate Hamiltonian dynamics for time* T *to obtain proposal at* $\boldsymbol{x}\,(T)$

Figure 3: HAMILTONIAN MONTE CARLO*: Given the current sample position* $\boldsymbol{x}\,(0)$ *and a target function (the 2D slice from Figure 2), a physical analogy of Hamiltonian Monte Carlo is:* ***(a)*** *first it takes the logarithm of the target function and flips it upside down so that "gravity" pulls towards high contribution areas.* ***(b)*** *Then it gives the current sample an initial momentum* $\boldsymbol{p}\,(0)$ *and* ***(c)*** *lets the point move for some time* T *with respect to the geometry of the flipped function.*



(a) *original target function and sample position*

(b) *simulate Hamiltonian dynamics on local quadratic approximation*

(c) *proposals result in a Gaussian distribution*

(d) *multiply with a prior to limit variance and draw proposals from red*

Figure 4: HESSIAN-HAMILTONIAN MONTE CARLO*:* ***(a)*** *Given the original function and sample position* $x\,(0)$*,* ***(b)*** *we approximate the costly Hamiltonian dynamics simulation by first constructing a local quadratic approximation at the current sample position* $\boldsymbol{x}\,(0)$*.* ***(c)*** *Different initial momentum* $\boldsymbol{p}\,(0)$ *results in different proposal positions* $\boldsymbol{x}\,(T)$*, which makes* $\boldsymbol{x}\,(T)$ *a random variable. We show that trajectories with a Gaussian PDF for initial momentum result in a Gaussian PDF (the yellow shaded area) for final destination.* ***(d)*** *Finally, we multiply the Gaussian with a prior (the purple shaded area) to prevent proposals from going too far when the second derivative is low. The resulting sample proposal distribution is shown in red, and we draw our proposals from the resulting distribution.*

Derivatives in rendering. Shinya *et al.* [1987] used a second-order power series along with paraxial approximation to approximate the neighborhood of a ray. Irradiance caching techniques [Ward et al. 1988; Ward and Heckbert 1992; Schwarzhaupt et al. 2012] compute the gradients and the Hessians of the irradiance with respect to the screen coordinates for sparse interpolation for diffuse or low-glossy surfaces. Ray differentials [Igehy 1999] and path differentials [Suykens and Willems 2001] compute the footprint of the light paths for texture filtering using first derivatives. Chen and Arvo [2000] use first and second-order derivatives of the specular light paths for sparse interpolation. Path gradients [Suykens and Willems 2001] are used for hierarchical radiosity applications, where the gradients of the paths are hand-derived. Ramamoorthi *et al.* [2007] performed a first-order analysis for direct illumination light field. Gradient-domain rendering approaches, e.g. [Lehtinen et al. 2013; Kettunen et al. 2015], sample in the gradient domain to exploit the sparsity of gradients in image space. They use finite differences of the path on the image coordinates, whereas our method uses analytical derivatives on all dimensions. While finite differences could capture the discontinuities of the signal, it is more expensive to generate and does not scale well with dimensionality.

Our usage of derivatives differs from previous works in several respects. First, we use automatic differentiation to compute the derivatives, which means that we do not assume any particular effect. This enables our method to be able to handle various combinations of lighting scenarios. Second, we take the derivatives with respect to all the sampling dimensions, so we can capture the high-dimensional structure of the light path. Third, we take both the first and the second derivatives. The Hessians enable us to take the correlation between the dimensions of the sampling domain into account. Finally, we apply the derivatives in the MCMC sampling context.

Automatic differentiation. Given a sequence of arithmetic operations, automatic differentiation [Griewank and Walther 2008] (AD) generates the derivatives of the program by applying chain rules to the computational graph of the program. Automatic differentiation is different from symbolic differentiation, in that it focuses on reusing the expressions by carefully traversing the computational graph, making it more efficient.

Automatic differentiation has been used in computer graphics occasionally [Grinspun et al. 2003; Piponi 2004; Guenter 2007] and was used in shader programming for ray footprint estimation [Gritz et al. 2010]. Manually deriving the derivatives is tedious and error-prone, and AD is a powerful tool that makes it possible for us to take the derivatives of any kind of light path.

3 Hamiltonian Monte Carlo

We review the Hamiltonian Monte Carlo (HMC) [Duane et al. 1987] method; see also Neal [2010] for a more thorough description and survey. Hamiltonian Monte Carlo is a variant of the Markov Chain Monte Carlo (MCMC) methods. HMC generates the new proposal samples from the current sample by simulating Hamiltonian dynamics driven by the landscape of the target function.

Recall that MCMC methods generate a sequence of samples $\boldsymbol{x}_i$, whose distribution converges to a distribution proportional to a specific target function $f^*(\boldsymbol{x}_i)$, by forming a Markov chain of the sample sequence. For the sake of notational simplicity, we denote the target function as $f(\boldsymbol{x})$. At iteration $i+1$, a new proposal sample is drawn from a distribution based on $\boldsymbol{x}_i$. Then the proposal sample is probabilistically accepted or rejected. If accepted, it forms the new state of the Markov chain. From now on, we assume that the samples $\boldsymbol{x}_i$ lie in a hypercube of $[0,1]^N$, similar to the primary sample space [Kelemen et al. 2002]. Operating directly on path-space [Veach 1998] is more challenging due to its definition as a cross product of lower-dimensional manifolds.

Figure 3 gives an illustration of Hamiltonian Monte Carlo where, in a nutshell, state is modified by giving the current sample a random initial velocity (or more precisely, a momentum), and simulating its motion under gravity. The target function first needs to be "flipped" so that high contribution regions correspond to a lower height and samples are attracted there by gravity (Figure 3 (b)). The particle is given an initial momentum, typically drawn from a Gaussian, and its motion is simulated in the height field given by the flipped contribution function for a fixed amount of time. Acceptance rules are then applied, although if the integrator preserves energy, samples are always accepted. This approach helps the samples stay in the high contribution region (low height in the flipped function) because of the effect of gravity.

Hamiltonian dynamics. Formally, Hamiltonian dynamics is a system of differential equations defined on the Hamiltonian energy E:

$$\begin{aligned}\frac{\partial \boldsymbol{x}}{\partial t} &= \frac{\partial E}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{p}}{\partial t} &= -\frac{\partial E}{\partial \boldsymbol{x}}.\end{aligned} \tag{6}$$

The auxiliary momentum variable $\boldsymbol{p}$ is introduced to drive the sampling of position $\boldsymbol{x}$, and $\boldsymbol{p}$ has the same number of dimensions as $\boldsymbol{x}$. The Hamiltonian energy $E(\boldsymbol{x},\boldsymbol{p})$ is a composite of the potential energy $U(\boldsymbol{x}) = -\log f(\boldsymbol{x})$ and the kinetic energy $K(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T A \boldsymbol{p}$. The potential energy is defined in the logarithmic domain to better capture the dynamic range of the target functions:

$$E(\boldsymbol{x},\boldsymbol{p}) = U(\boldsymbol{x}) + K(\boldsymbol{p}) = -\log f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{p}^T A \boldsymbol{p}, \tag{7}$$

where A is a user-defined "inverse mass matrix", which represents the inverse of the mass of the particle. Typically, it is set to a scalar $\frac{1}{m}$ times an identity matrix, where m is the mass, but in our work we will use a full matrix (discussed in Section 4). The negative of the function $\log f(x)$ is taken to enable high contribution regions to have low potential energy (as shown in Figure 3).

We substitute the definition of the Hamiltonian energy (Equation (7)) into the Hamiltonian equation (Equation (6)), and obtain:

$$\begin{aligned}\frac{\partial \boldsymbol{x}}{\partial t} &= A\boldsymbol{p} \\ \frac{\partial \boldsymbol{p}}{\partial t} &= \frac{\partial \log f(\boldsymbol{x})}{\partial \boldsymbol{x}}.\end{aligned} \tag{8}$$

Equation (8) defines a trajectory of position $\boldsymbol{x}$ and momentum $\boldsymbol{p}$ over time t. Intuitively, if the momentum at time t is high, we will make a large jump from the current position $\boldsymbol{x}(t)$, and if the derivatives of the target function at $\boldsymbol{x}(t)$ are low, the increment to the momentum will be small. Hamiltonian Monte Carlo is highly adaptive to the local structure of the target function.

MCMC with Hamiltonian dynamics. To apply Hamiltonian dynamics in the context of MCMC, we first take the exponent of the negative Hamiltonian energy:

$$\exp(-E(\boldsymbol{x},\boldsymbol{p})) = f(\boldsymbol{x}) \exp\left(-\frac{1}{2}\boldsymbol{p}^T A \boldsymbol{p}\right) =: f(\boldsymbol{x})\phi(\boldsymbol{p}). \tag{9}$$

$\exp\left(-\frac{1}{2}\boldsymbol{p}^T A \boldsymbol{p}\right)$, which we denote as $\phi(\boldsymbol{p})$, is proportional to the PDF of a zero-mean Gaussian with covariance A^{-1}. To generate a new proposal position, we pick a zero-mean Gaussian distributed momentum $\boldsymbol{p}(0)$ with covariance A^{-1} and a fixed time T, and simulate the Hamiltonian dynamics to obtain the position at $\boldsymbol{x}(T)$.

The proposal position is probabilistically accepted with the probability $a((\boldsymbol{x}(0),\boldsymbol{p}(0)) \to (\boldsymbol{x}(T),\boldsymbol{p}(T)))$, where

$$\begin{aligned}&a((\boldsymbol{x}(0),\boldsymbol{p}(0)) \to (\boldsymbol{x}(T),\boldsymbol{p}(T))) \\ &\quad= \min\left(\frac{\exp(-E(\boldsymbol{x}(T),\boldsymbol{p}(T)))}{\exp(-E(\boldsymbol{x}(0),\boldsymbol{p}(0)))}, 1\right) \\ &\quad= \min\left(\frac{f(\boldsymbol{x}(T))\,\phi(\boldsymbol{p}(T))}{f(\boldsymbol{x}(0))\,\phi(\boldsymbol{p}(0))}, 1\right).\end{aligned} \tag{10}$$

Intuitively, the acceptance rule resembles the Metropolis-Hastings rule (Equations (2) and (3)), with the transition probability Q substituted with the (unnormalized) PDF of the momentum Gaussian ϕ. Furthermore, if the Hamiltonian dynamics is simulated perfectly, $\exp(-E(\boldsymbol{x},\boldsymbol{p}))$ is a constant throughout the simulation because of energy conservation, and the acceptance probability is 1.

Properties of Hamiltonian dynamics. More formally, given a fixed time T, the Hamiltonian equation creates a mapping M between $(\boldsymbol{x}(0),\boldsymbol{p}(0))$ and $(\boldsymbol{x}(T),\boldsymbol{p}(T))$. Neal [2010] showed that this mapping has several important properties:

- The mapping is time-reversible: if we flip the momentum at time T and use $(\boldsymbol{x}(T),-\boldsymbol{p}(T))$ as the input to M, the output of the mapping would be $(\boldsymbol{x}(0),-\boldsymbol{p}(0))$. That is, if we simulate the Hamiltonian dynamics in a backward manner from the end point, it will go back to the starting point.
- The mapping preserves the volume: If we apply the mapping for a region R_0 of points $(\boldsymbol{x}(0),\boldsymbol{p}(0))$, and map them to another region R_T, the volumes of the two regions in the position-momentum space remain the same (known as Liouville's theorem).
- The mapping preserves energy: the Hamiltonian energy E (Equation (7)) remains the same after the mapping.

The first property is crucial for the detailed balance condition (Equation (4)) to hold, since it ensures that the mapping is one-to-one. The second property ensures that we do not need to account for the Jacobian of the mapping in the Metropolis acceptance rule. The energy preservation property shows that the probability of acceptance is in fact 1 since $E(\boldsymbol{x}(0),\boldsymbol{p}(0)) = E(\boldsymbol{x}(T),\boldsymbol{p}(T))$.

Unfortunately, Equation (8) does not have a known analytical solution for an arbitrary target function. It is usually required to integrate the differential equation using numerical integrators such as leapfrog integrators. These integrators maintain the time-reversibility and volume-preservation, but do not preserve energy. The Hamiltonian

dynamics are approximated and the acceptance probability is no longer 1. Furthermore, numerical integrators are expensive for light transport simulation because each step involves costly ray tracing operations and derivative computations of the shader.

Discussion. Hamiltonian Monte Carlo has been used in graphics recently for computational design [Ritchie et al. 2015]. The Metropolis-adjusted Langevin algorithm (MALA) [Roberts and Tweedie 1996] is a one-step approximation to Hamiltonian Monte Carlo. MALA makes the proposal distribution isotropic, except that the mean of the proposal distribution is shifted by the first derivatives (gradient) times a user-specified constant. Our method is also a one-step approximation, but the proposal distribution of our method adapts to the anisotropy of the signal, because we utilize the second derivatives. It is possible to precondition the MALA algorithm using a positive-definite mass matrix, such as the Fisher information matrix [Girolami and Calderhead 2011]. However, it remains unclear how to relate the Hessian matrix to the positive-definite mass matrix. Betancourt [2013] proposed a SOFTABS metric that removes the sign of the eigenvalues of the Hessian matrix using a smooth mapping. In contrast, we treat positive and negative eigenvalues differently by directly simulating Hamiltonian dynamics on the quadratic landscape.

4 Hessian-Hamiltonian Monte Carlo

Figure 4 provides some intuition of our sampling algorithm. We compute the second order Taylor expansion (local quadratic approximation) of the logarithm of the target function first, where the gradient and the Hessian are computed using automatic differentiation. The quadratic function does not define a proper distribution, since it might grow to infinity, which prevents us from directly importance sampling it. Hamiltonian dynamics enables us to sample from this quadratic function to obtain the proposal position, since it works on any continuous function.

The Hamiltonian dynamics have an analytical solution in the case of a quadratic function. However, we cannot use the acceptance rule in standard Hamiltonian Monte Carlo (Equation (10)) to compute the acceptance probability. It would break time-reversibility, since each light path would have a different associated quadratic function. Fortunately, we can derive from the analytical solution, that the distribution of a proposal, given a Gaussian momentum, is a Gaussian distribution (Figure 4 (c)). Therefore, we associate each light path with a Gaussian distribution derived from the quadratic function and Hamiltonian dynamics, and it is possible to compute the acceptance probability using the Metropolis-Hastings rule (Equation (3)). Finally, we multiply the analytical Gaussian with a prior Gaussian distribution to place a limit on its variance (Figure 4 (d)), so that the proposals do not go too far away where the second order approximation can be inaccurate.

Approximating Hamiltonian dynamics. We first show how to derive the closed-form solution to the differential equations for Hamiltonian dynamics (Equation (8)), given an initial momentum and position. Then, we will show how to infer the Gaussian distribution of proposals. We start from a second-order approximation of $\log f$. For the sake of simplicity and without loss of generality, in the following we assume the current position $\boldsymbol{x}(0)$ is at the origin. Any small offset $\boldsymbol{x}$ from the origin can be approximated by:

$$\log f(\boldsymbol{x}) \approx \frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x} + G^T \boldsymbol{x} + \log f(0), \tag{11}$$

where H is the Hessian matrix and G is the gradient vector at $\log f(\mathbf{0})$. If we substitute this approximation into the Hamiltonian equation (Equation (8)) using $\frac{\partial \log f(\boldsymbol{x})}{\partial \boldsymbol{x}} \approx H\boldsymbol{x} + G$ and combine the two differential equations, we get:

$$\frac{\partial^2 \boldsymbol{x}(t)}{\partial t^2} = AH\boldsymbol{x}(t) + AG. \tag{12}$$

The above equation is a standard second-order differential equation system, and has an analytical solution. We start from the one-dimensional case, then generalize it to higher dimensions. Assuming x is a one-dimensional variable, if we let $\alpha = AH, \beta = AG$, an analytical solution is:

$$x(t) = \begin{cases} c_1 \exp\left(\sqrt{\alpha}t\right) + c_2 \exp\left(-\sqrt{\alpha}t\right) - \frac{\beta}{\alpha} & \text{if } \alpha > 0 \\ c_1 \cos\left(\sqrt{-\alpha}t\right) + c_2 \sin\left(\sqrt{-\alpha}t\right) - \frac{\beta}{\alpha} & \text{if } \alpha < 0 \\ c_1 t + c_2 + \frac{\beta t^2}{2} & \text{if } \alpha = 0, \end{cases} \tag{13}$$

which can be verified by plugging the solution back into the equation. The constant multipliers c_1, c_2 can be obtained by plugging in the initial condition $x(0) = 0$, $x'(0) = Ap(0)$ (where $p(0)$ is sampled from the Gaussian distribution ϕ defined in Equation (9)) into the original Hamiltonian equation (Equation (8)). Specifically, the constants are:

$$\begin{aligned} c_1 &= \begin{cases} \frac{1}{2}\left(\frac{\beta}{\alpha} + \frac{\hat{p}(0)}{\sqrt{\alpha}}\right) & \text{if } \alpha > 0 \\ \frac{\beta}{\alpha} & \text{if } \alpha < 0 \\ \hat{p}(0) & \text{if } \alpha = 0 \end{cases} \\ c_2 &= \begin{cases} \frac{1}{2}\left(\frac{\beta}{\alpha} - \frac{\hat{p}(0)}{\sqrt{\alpha}}\right) & \text{if } \alpha > 0 \\ \frac{\hat{p}(0)}{\sqrt{-\alpha}} & \text{if } \alpha < 0 \\ 0 & \text{if } \alpha = 0, \end{cases} \end{aligned} \tag{14}$$

where we denote $\hat{p}(0) = Ap(0)$ to simplify the equation.

To illustrate, since the inverse mass A is required to be positive, if the second derivative H is strictly negative, we consider the $\alpha < 0$ case and the trajectory $x(t)$ becomes a linear combination of a cosine curve and a sine curve, which oscillates in the ridges of the flipped function. On the other hand, if the second derivative is strictly positive, then the trajectory climbs straight up the hill and goes to infinity as t increases.

If x is an N-dimensional vector instead, the general solution of this differential equation system becomes a linear combination of the eigenvectors $\boldsymbol{e}_i$ of the matrix AH:

$$\boldsymbol{x}(t) = \sum_{i=1}^{N} x_i(t)\,\boldsymbol{e}_i, \tag{15}$$

where the coefficient $x_i(t)$ is similar to the one-dimensional case (Equation (13)), but with α substituted with matrix AH's i-th eigenvalue λ_i, and β and $\hat{p}(0)$ substituted with the projection of the vector AG and $\hat{\boldsymbol{p}}(0)$ on the i-th eigenvector $\boldsymbol{e}_i$, respectively. Again, we can obtain the constant multipliers as in the one-dimensional case by plugging in the initial conditions.

A Gaussian equivalent to the approximation. We have derived an analytical trajectory for a fixed initial momentum. However, having the analytical trajectory is not enough. Recall that Hamiltonian Monte Carlo starts by generating a Gaussian distributed momentum $\boldsymbol{p}(0) \sim \mathcal{N}\left(0, A^{-1}\right)$, and generates a new position proposal $x(T)$ at a fixed time T. Unfortunately, a direct application of the analytical solution to Hamiltonian Monte Carlo using the original acceptance rule (Equation (10)) is infeasible. The gradient and Hessian generally would be different at the proposal position, and the time-reversibility would be violated.

(a) $\sigma^2 = 0.028$ *accept rate* 28.96% **(b)** $\sigma^2 = 0.007$ *accept rate* 54.02% **(c)** $\sigma^2 = 0.001$ *accept rate* 82.11%

Figure 5: *We show the effect of the prior Gaussian parameter σ^2 using an inset from the* BATHROOM *scene (Figure 8).* ***(a)*** *High σ^2 results in low acceptance rate and a noisy image.* ***(c)*** *Low σ^2 results in high acceptance rate, but produces correlated noise.* ***(b)*** *We choose a σ^2 so that the acceptance rate falls in the ranges from $50\% - 70\%$.*

An observation from the analytical solution (Equations (13) and (14)), is that the Hamiltonian dynamics are actually linear mappings from the Gaussian distributed variable $\boldsymbol{p}(0)$ to the new position $\boldsymbol{x}(T)$ if we have $t = T$ fixed. This means that $\boldsymbol{x}(T)$ is also Gaussian distributed since Gaussian variables are closed under linear transform. Therefore, we can generate $\boldsymbol{x}(T)$ using *a single Gaussian distribution*. Furthermore, the PDF of the Gaussian can be used as the transition probability Q to compute the Metropolis-Hastings acceptance probability (Equation (3)).

Now we will show why the mapping is linear and how to derive the covariance and the mean of $\boldsymbol{x}(T)$. Again we start from the one-dimensional case. If we plug the multipliers c_1 and c_2 (Equation (14)) into the analytical solution (Equation (13)) and rearrange the terms in one-dimensional $x(T)$, we have:

$$\begin{aligned} x(T) &= \begin{cases} \left(\frac{\exp\left(\sqrt{\alpha}T\right)-\exp\left(-\sqrt{\alpha}T\right)}{2\sqrt{\alpha}}\right)\hat{p}(0) & \\ \quad +\frac{\beta}{2\alpha}\left(\exp\left(\sqrt{\alpha}T\right)+\exp\left(-\sqrt{\alpha}T\right)-1\right) & \text{if } \alpha > 0 \\ \frac{1}{\sqrt{-\alpha}}\sin\left(\sqrt{-\alpha}T\right)\hat{p}(0) & \\ \quad +\frac{\beta}{\alpha}\left(\cos\left(\sqrt{-\alpha}T\right)-1\right) & \text{if } \alpha < 0 \\ T\hat{p}(0)+\frac{\beta T^2}{2} & \text{if } \alpha = 0 \end{cases} \\ &= s\hat{p}(0) + o \\ &= sAp(0) + o, \end{aligned} \tag{16}$$

which is a linear function of $p(0)$ and we denote the scaling coefficient as s and the offset coefficient as o.

For the N-dimensional case, since $\boldsymbol{x}(T)$ is a linear combination of $x_i(T)$ (Equation (15)), it is still a linear transform. Moreover, if we write $x_i(T) = s_i \cdot \hat{p}_i(0) + o_i$, where $\hat{p}_i(0)$ is the projection of $\hat{\boldsymbol{p}}(0)$ on the i-th eigenvector $\boldsymbol{e}_i$ of the matrix AH, we can write out the linear transformation in matrix form:

$$\boldsymbol{x}(T) = SA\boldsymbol{p}(0) + \boldsymbol{o}, \tag{17}$$

where the matrix S and the vector $\boldsymbol{o}$ can be obtained from the eigenvectors $\boldsymbol{e}_i$, and the coefficients s_i and o_i:

$$S = \sum_{i=1}^{N} s_i \boldsymbol{e}_i, \quad \boldsymbol{o} = \sum_{i=1}^{N} o_i \boldsymbol{e}_i. \tag{18}$$

Recall that $\boldsymbol{p}(0)$ is a zero-mean Gaussian variable with covariance A^{-1}. Therefore the covariance matrix Σ and the mean $\boldsymbol{\mu}$ of the Gaussian random variable $\boldsymbol{x}(T)$ are

$$\Sigma = (SA)A^{-1}(SA)^T = SAS^T, \quad \boldsymbol{\mu} = \boldsymbol{o}. \tag{19}$$

Multiplying with prior Gaussian. In practice, our second order approximation (Equation (11)) can be inaccurate when the proposal is far from the current state, or if there are discontinuities such as visibility changes. To compensate for this, we introduce a prior Gaussian distribution with a zero mean and isotropic variance using a user specified constant σ^2, and multiply the PDF of it with the PDF of the Gaussian random variable $\boldsymbol{x}(T)$, to effectively place a limit on the maximum variance (which corresponds to the movement of the path in path space).

Another way to think about the prior is that it acts as a regularization term that penalizes high variance. If σ^2 is high, then the change of the light path would be large, and the acceptance rate would be lower. On the other hand, if σ^2 is low, then the change of the light path is small, and the acceptance rate would be higher. We show the effects of different σ^2 in Figure 5. In our current implementation we manually set σ^2 to achieve a certain acceptance rate (50% to 70%), but it may be possible to automatically adjust the parameter using adaptive MCMC [Andrieu and Thoms 2008]. The final mean $\boldsymbol{\mu}^*$ and covariance Σ^* are

$$\Sigma^* = \left(\Sigma^{-1} + \frac{1}{\sigma^2}\right)^{-1}, \quad \boldsymbol{\mu}^* = \Sigma^*\Sigma^{-1}\boldsymbol{o}. \tag{20}$$

Computing acceptance probability In order to apply the Metropolis-Hastings rule (Equation (2)) given a current position $\boldsymbol{x}$, we generate a new proposal position $\boldsymbol{y}$ from a Gaussian variable with mean $\boldsymbol{\mu}_x^*$ and covariance matrix Σ_x^* computed using Equation (20). Then we compute the mean $\boldsymbol{\mu}_y^*$ and covariance matrix Σ_y^* at the proposal position. The acceptance probability (Equation (3)) is computed using the PDFs of the Gaussians:

$$\begin{aligned} a(\boldsymbol{x} \to \boldsymbol{y}) &= \min\left(1, \frac{f(\boldsymbol{y})\,Q(\boldsymbol{y} \to \boldsymbol{x})}{f(\boldsymbol{x})\,Q(\boldsymbol{x} \to \boldsymbol{y})}\right) \\ &= \min\left(1, \frac{f(\boldsymbol{y})\,\Phi_{\boldsymbol{y}}(\boldsymbol{x}-\boldsymbol{y})}{f(\boldsymbol{x})\,\Phi_{\boldsymbol{x}}(\boldsymbol{y}-\boldsymbol{x})}\right), \end{aligned} \tag{21}$$

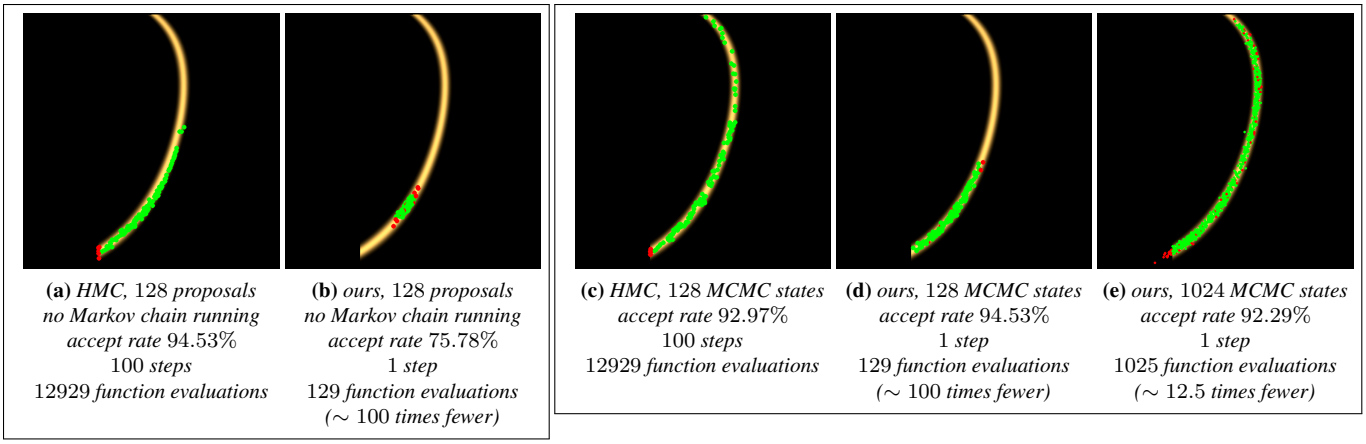
where $\Phi_{\boldsymbol{x}}(\boldsymbol{y}-\boldsymbol{x})$ is the Gaussian PDF with covariance Σ_x^* and mean $\boldsymbol{\mu}_x^*$ computed at $\boldsymbol{x}$ (Equation (20)). Specifically, if we define $\boldsymbol{z} = \boldsymbol{y} - \boldsymbol{x}$, it is:

$$\Phi_{\boldsymbol{x}}(\boldsymbol{z}) = (2\pi)^{-\frac{N}{2}} |\Sigma_x^*|^{\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu}_x^*)^T \Sigma_x^{*-1}(\boldsymbol{z}-\boldsymbol{\mu}_x^*)\right). \tag{22}$$

$\Phi_{\boldsymbol{y}}(-\boldsymbol{z})$ is defined similarly with covariance and mean computed at $\boldsymbol{y}$.

Setting parameters A and T. A remaining question is how to choose the inverse mass matrix A and simulation time T. Previous work in Hamiltonian Monte Carlo suggests setting A to the covariance of the target function [Neal 2010; Girolami and Calderhead 2011]. As an example, consider a target function $f(\mathbf{x})$ that is a Gaussian distribution with covariance Σ_f. If we ignore multiplication with the prior in Section 4.3, setting A to the covariance of the target function, and setting $T = \pi/2$ will result in a Gaussian mutation (Equation (19)) that precisely matches the target function. [1]

[1] In this case, the Hessian H for $\log f$ will simply be $-\Sigma_f^{-1}$, and $\alpha = AH$ will be a negative identity matrix. Therefore, we consider the $\alpha < 0$ case in Equation (16), where $s = 1$ for $T = \pi/2$, and S is the identity matrix. Therefore, the covariance matrix Σ from Equation (19) is given simply by A, leading to a Gaussian distribution with covariance Σ_f, which is exactly the target function. This justifies setting A to the covariance of the target function, and setting $T = \pi/2$.



(a) *HMC,* 128 *proposals*
no Markov chain running
accept rate 94.53%
100 *steps*
12929 *function evaluations*

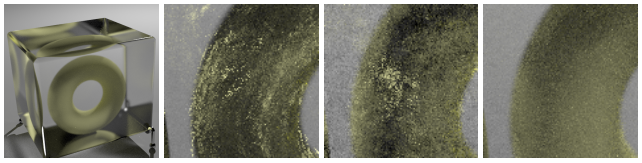
(b) *ours,* 128 *proposals*
no Markov chain running
accept rate 75.78%
1 *step*
129 *function evaluations*
(∼ 100 *times fewer)*

(c) *HMC,* 128 *MCMC states*
accept rate 92.97%
100 *steps*
12929 *function evaluations*

(d) *ours,* 128 *MCMC states*
accept rate 94.53%
1 *step*
129 *function evaluations*
(∼ 100 *times fewer)*

(e) *ours,* 1024 *MCMC states*
accept rate 92.29%
1 *step*
1025 *function evaluations*
(∼ 12.5 *times fewer)*

Figure 6: *We compare the sample distribution of the original HMC method and our method using the zoomed out slices from Figures 2* ***(g)*** *and* ***(h)****. The left box shows the "proposals" drawn from the current sample position, without running the Markov chain, and the right box shows the actual Markov chain states. While the original HMC is able to generate proposals with high acceptance probability, and over longer trajectories (compare* ***(a)*** *to* ***(b)*** *and* ***(c)*** *to* ***(d)****), each proposal in the original HMC requires many steps to compute (*100 *steps in this case), and each step involves costly ray tracing, shading, and derivative computation. Our method can achieve much better space coverage using a single step (as opposed to the* 100 *steps in the original HMC), and requires an order of magnitude fewer function evaluations* ***(e)****.*



(a) TORUS

(b) *Kelemen et al., isotropic proposal*

(c) *Ours, isotropic proposal*

(d) *Ours,* H^2MC *proposal*

Figure 7: *We show a comparison between our new parameterization and Kelemen et al.'s parameterization on the* TORUS *scene with a diffuse torus inside a glossy glass cube lit by a point light. The left image is computed using 5000 samples per pixel using our method, and the three insets are computed with 256 samples per pixel. The original parameterization incurs correlation between screen space and the outgoing sample directions on the glass and the torus, creating streaks on the torus. Our new parameterization greatly reduces this correlation. Our Hessian-Hamiltonian proposal further improves the sampling efficiency dramatically.*

In general, the target function need not be a Gaussian and a global covariance Σ_f may not be sufficient to describe the function. We approximate the covariance locally using the fact that we have the Hessian H of the log of the function. If the target function is a Gaussian, the negative inverse of the Hessian $-H^{-1}$ would exactly be the covariance of the target function. It would be tempting to directly set A to $-H^{-1}$, but the covariance matrix of a Gaussian distribution is required to be positive semidefinite (all eigenvalues need to be positive), and $-H^{-1}$ is not necessarily positive definite in general. We approximate the local covariance of the function by substituting the eigenvalues in $-H^{-1}$ by their absolute values, and set A to the approximated local covariance:

$$A = \sum_{i=1}^{N} \begin{cases} \frac{1}{|\lambda_i^H|} \boldsymbol{e}_i^H & \text{if } \lambda_i^H \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

where $\boldsymbol{e}_i^H$ and λ_i^H are the i-th eigenvector and eigenvalue of H. Finally, we set T to $\frac{\pi}{2}$, as in the Gaussian example above.

The construction of A and T also simplifies the implementation, since A and H share the same set of eigenvectors and A's eigenvalues are the inverse of the absolute value of H's eigenvalues or zero. The eigenvalues λ_i of matrix AH would then be either -1, 1, or 0, depending on the sign of the eigenvalue of H. The magnitudes of the eigenvalues in the Hessian H (and hence A) are still taken into consideration when sampling from the momentum using the inverse mass matrix A. We show the pseudo-code of our algorithm in Appendix A, which outputs the final mean $\boldsymbol{\mu}^*$ and covariance Σ^* given the gradient G, Hessian H, and prior σ^2.

Finally, we compare the proposals and the Markov chain of original HMC with a leapfrog integrator, and our Hessian-HMC method in Figure 6, using a 2D slice in the RING scene (Figure 2). We use a step size of 0.0005 with 100 steps for the leapfrog numerical integrator in HMC, and we set the prior Gaussian $\sigma^2 = 0.01$ for our method. The target acceptance rate is set higher because the dimensionality of the function is low [Neal 2010]. Although original HMC is able to use longer trajectories to explore the space more thoroughly with the same number of samples, a single sample in HMC requires 100 steps of ray tracing, shading and derivatives computation. Choosing a bigger step size or smaller step number for HMC may result in energy loss or inferior space exploration efficiency, and this parameter of original HMC is notoriously hard to tune. Our H^2MC method can explore the space better using an order of magnitude fewer function evaluations (Figure 6(e)).

5 Implementation

We implement our method in a stand-alone MCMC renderer with the Embree ray tracing engine [Wald et al. 2014]. We use the automatic differentiation library CppAD [Bell 2003–2015] and CppADCodeGen [Leal 2011–2015] to generate the code for the derivatives of the path throughput function. The renderer supports the Phong BRDF, the microfacet refraction model [Walter et al. 2007], point and area light sources, and linear object motion. Each sample in the Markov chain represents a single light path that connects the light to the camera. As in most previous MCMC rendering methods, we employ multiple mutation strategies to better cover different types of light paths. Specifically, we adopt three different types of mutation strategies: a multiplexed *large step* mutation [Kelemen et al. 2002; Hachisuka et al. 2014], a novel modified *small step* perturbation, and a lens perturbation. The large step mutation is responsible for making large jumps between different disconnected components of light paths, the small step perturbation is responsible for making a small change to all dimensions of the function, and finally the lens perturbation changes only part of the light path to alleviate difficult visibility issues. We apply the H^2MC sampling on the small step

and the lens perturbation to explore the local structure of the path throughput function. In the rest of this section, we address some technical details of the implementation.

Multiplexed large step mutation. To ensure the ergodicity of the Markov chain, that is, to ensure we have a strictly positive probability to sample all light paths with non-zero contribution, we include a large step mutation to generate a proposal light path that is completely independent of the current sample. Furthermore, the generation of the proposal is *multiplexed* in the same spirit as the Multiplexed Metropolis Light Transport [Hachisuka et al. 2014]. Specifically, a light path is generated by first choosing a path length (the distribution is determined during the initialization), then we uniformly sample the length of the camera subpath. The camera subpath and the light subpath are sampled correspondingly and their endpoints are connected. The multiple importance sampling weight is computed as usual and the path contribution is scaled by the number of techniques. Since the generation of each such path is independent, H^2MC sampling is not applied in this mutation.

H^2 small step perturbation. We adopt a modified version of the *small step* perturbation [Kelemen et al. 2002] as the main component to explore the path throughput function locally. In Kelemen *et al.*'s work, the light paths are represented as the random numbers that are used to generate them. The perturbation is done by making small changes to the random numbers, and results in a new light path. We make two modifications to the parameterization.

First, we classify the surfaces into specular and non-specular by applying a user-defined threshold on the roughness. If the surface is near-specular, the outgoing directions are parameterized using the random numbers. On the other hand, if the surface is non-specular, the outgoing directions are parameterized using the global directions expressed in absolute spherical coordinates. We found that this change improves sampling efficiency because the correlation between the dimensions is reduced. Kelemen's parameterization handles specular surfaces well, because importance sampling captures the peak of the target function well. On the other hand, the local parameterization introduces extra correlation between dimensions, because the outgoing direction depends on the normal of the surface, and the normal depends on the previous outgoing direction. The parameterization change is beneficial because H^2MC captures the *linear* correlation between dimensions, while the parameterization change is non-linear. We show a comparison of the original parameterization with the new one in Figure 7.

Second, if the light path hits a light source without next event estimation (that is, no explicit connection is made), we substitute the parameterization of the last outgoing direction, to the position on the light source, so that the perturbation is more likely to hit the light source. We assume a pinhole camera in our implementation, but the second change can also apply in the case when the light path starts from the light source and hits the camera lens without explicit connection. The new parameterization represents the sample position $\boldsymbol{x}$ in the H^2MC sampling.

The time dimension is treated the same as other dimensions. The generality of H^2MC sampling makes it agnostic to the underlying representation. This enables us to detect the correlation between time and other dimensions, which was not considered in previous MCMC rendering methods.

H^2 lens perturbation. Consider light paths involving small and flat surfaces. If we mutate the whole path, chances are high that we will miss the surfaces and result in zero contribution. A better strategy for these light paths is to mutate only a subset of the full path, and keep the rest of the vertices fixed. We implement the lens perturbation in the original Metropolis Light Transport algorithm [Veach and Guibas 1997], which mutates only the lens subpath. For lens perturbation, the sample position $\boldsymbol{x}$ in the H^2MC sampling is the two dimensional image coordinate.

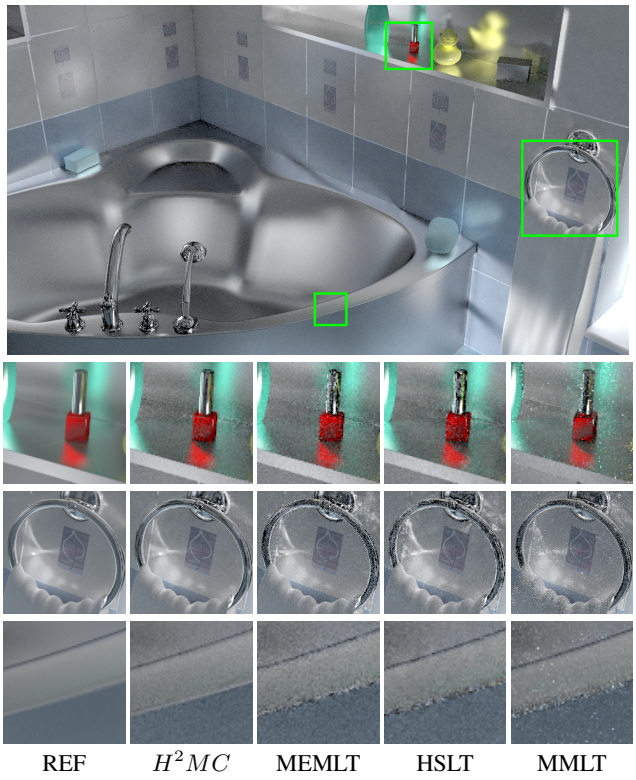


Figure 8: BATHROOM*: An equal-time (10 minutes) comparison on the bathroom scene with multiple glossy reflections lit by a distant area light. The top image is generated by our method in 10 minutes. Our method achieves less noisy results on highly curved glossy surfaces and the caustics because we can adapt to the curvatures of the surfaces using second-order derivatives.*

	H^2MC	MMLT	MEMLT	HSLT
BATHROOM	610	1288	600	331
KITCHEN	5169	12453	4749	3319
BALLS	2943	8554	2961	N/A
CARS	1576	5361	1422	N/A

Table 1: *Sample count per pixel of each method for the equal-time comparisons.*

6 Results and Discussion

We compare against three other MCMC rendering methods: Multiplexed MLT (MMLT) [Hachisuka et al. 2014], Manifold Exploration MLT (MEMLT) [Jakob and Marschner 2012], and the improved Half-vector Space Light Transport (HSLT) [Hanika et al. 2015]. MMLT is a general rendering algorithm that does not assume any particular lighting effect, but its isotropic mutation makes it inefficient on difficult light paths such as highly-glossy transports. We compare to MMLT to show the efficiency of the anisotropic proposal sampling. MEMLT and HSLT are two rendering algorithms dedicated to specular and glossy transport by using first-order derivatives of the half-vectors. They can efficiently resolve difficult specular light paths, but often produce noisy results on highly-curved surfaces.

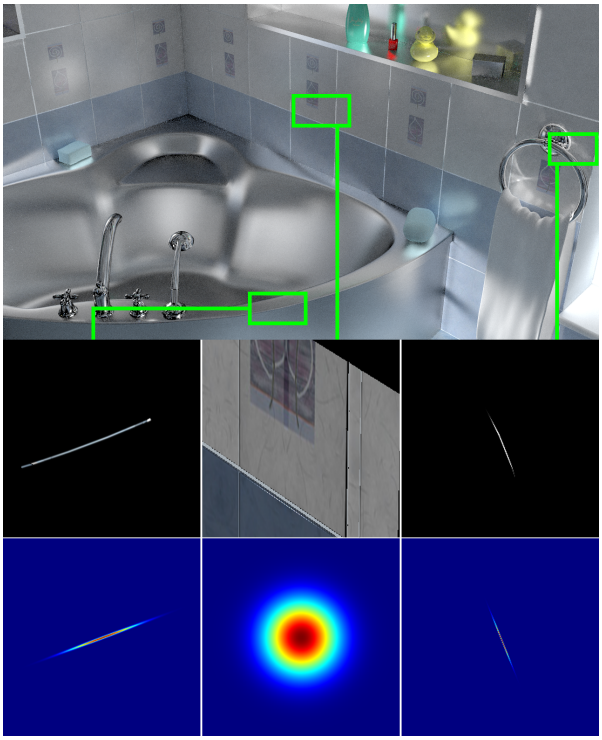


Figure 9: *We visualize the screen space slice of the contribution of three different light paths and our Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}^*, \Sigma^*)$ in the* BATHROOM *scene. The center row of the insets shows the contribution of perturbing the light path in the screen space. The left column shows a 4 bounce glossy reflection light path, the center column shows a 3 bounce diffuse reflection light path, and the right column shows a 3 bounce caustic light path caused by the metal towel ring. Glossy/specular transport results in sparse and anisotropic contributions, which are hard to sample using isotropic mutations. The bottom row shows our Gaussian approximation projected onto the screen space. The approximation matches the sharp contribution function and falls back to isotropic sampling when the contribution is smooth. Note that our method is anisotropic in all sampling dimensions, and we only show the screen space slice for visualization purposes.*

Furthermore, since they assume a specific lighting scenario, they cannot resolve difficult moving caustics, and usually result in ghosting artifacts (Figures 1 and 11). We did not compare to HSLT on the scenes with motion blur because their implementation does not allow it. We render four scenes – BATHROOM (1280×720), KITCHEN (1024×576), BALLS (768×576), CARS (768×576) – with different lighting, material, and geometry configurations (Figure 1 and Figures 8 to 11).

For MMLT we use our own implementation, for MEMLT and HSLT we use the implementation in the Mitsuba [Jakob 2010] renderer. HSLT is used with the lens perturbation because in our experiments it results in better images. The comparisons are equal-time using an Intel Core i7-4770 at 3.40GHz using 4 cores. The maximum path length is set to 7. References are rendered using the PSSMLT [Kelemen et al. 2002] implementation in Mitsuba and rendered for 2-3 days on a 64 core machine, except that the reference for the CARS is rendered using our method for roughly 15 hours on the 4 core machine (PSSMLT did not converge in 2-3 days computation). We show the sample count per pixel of each method in each scene in Table 1. In general our method is 2-3.5 times slower per sample than MMLT because of the derivatives and Gaussian computation, and is about the same speed as MEMLT. HSLT is slower than MEMLT

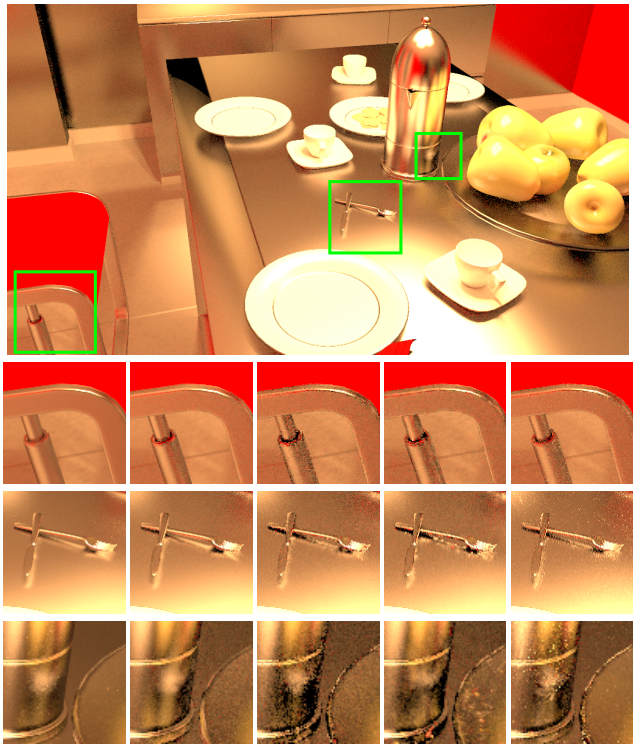


Figure 10: KITCHEN*: An equal-time (1 hour) comparison on the kitchen scene with complex material and geometry configuration lit by four area lights right above the table. The top image is generated by our method in an hour. This is a challenging scene and the reference rendered by PSSMLT is still slightly noisy after 2 days of computation on a 64 core machine. Our method excels at following the small features of the image such as the fork and the knife on the table, or the edges on the chair. It is also good at following the multiple glossy reflections on the highly curved surfaces such as the reflection on the flask.*

because it works on a higher-dimensional manifold.

Bathroom Figure 8 shows an equal-time (10 minutes) comparison on the *bathroom* scene with multiple glossy-to-glossy transports lit by a distant area light. For this particular scene, only indirect illumination is shown to highlight the differences between the algorithms. MMLT generates noisy results because of their isotropic mutation distribution. MEMLT and HSLT do generally well, but produce noisy results on high curvature surfaces because they use a first-order approximation on the surface. Our method is able to capture the local structure of the function and generates accurate results.

To demonstrate the anisotropic proposal distribution of our method, we visualize the screen space slice of the contribution of some light paths and the slice of our Gaussian approximation in Figure 9. Our method is able to adapt to the sparse and sharp path contribution function, and fall back to isotropic sampling when the contribution function is smooth. MEMLT and HSLT often fail to capture small screen space features, because they isotropically sample some dimensions first, and such sampling often misses the feature. Note that our method adapts to all dimensions, and we only show the screen space slice for the sake of visualization.

Kitchen. Figure 10 shows an equal-time (1 hour) comparison on the *kitchen* scene with complex materials and a difficult geometry

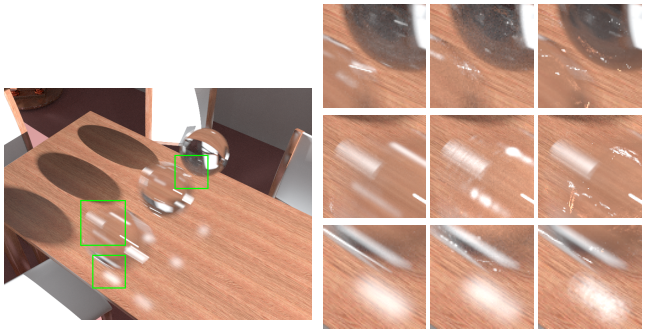


Figure 11: BALLS*: 30 minute rendering of the balls scene, which consists of three moving near-specular glass balls lit by a point light. The left image is generated by our method in 30 minutes. The moving balls show complex patterns with a combination of reflection from the room and the resulting caustics on the table. Neither MMLT nor MEMLT are able to efficiently resolve the moving features within the given time budget. Our method is able to closely follow the specular highlights and caustics in the glass because it detects the correlation between the time domain and path-space.*

configuration lit by four area lights close to the table. This is a challenging scene and the reference rendered by PSSMLT is still slightly noisy after 2 days of computation on a 64 core machine. MMLT produces spiky noise because some glossy-to-glossy light paths have small and high-contribution regions. MEMLT and HSLT generate noisy results on small and highly curved surfaces. Our method is able to follow the small image features closely, producing smoother results.

In general, light paths involving highly curved surfaces can be troublesome for MEMLT and HSLT, which only utilize first derivatives. Both of them need to start from an initial subpath, then iteratively converge to the new light path on the manifold. The light paths involving curved surfaces often have narrow contribution areas, and are highly non-linear. It is likely that the initial subpath will miss the highlight entirely, making it impossible to converge to a new light path. Even if the initial subpath hits the highlight, it could take many iterations to converge due to the non-linearity. In contrast, the second derivatives along with the Hamiltonian dynamics enable us to generate the proposal path directly with respect to the local shape of the function, avoiding the convergence issue.

Balls. Figure 11 shows a 30 minute rendering of the *balls* scene, which consists of three moving near-specular glass balls lit by a point light. MMLT is unable to resolve the difficult specular-diffuse-specular paths inside the moving balls and the caustics on the table. While MEMLT excels at resolving the specular light paths given the time fixed, it relies on seeding to sample the time dimension, which causes the ghosting artifacts on the balls. Our method is able to capture correlation between the time and the path-space, so that it can efficiently sample the difficult moving caustics and specular highlights.

Cars. Figure 1 shows a 20 minute rendering of the *cars* scene, with a static car and a moving car lit by an area light. This is a challenging scene because of the hard-to-find specular-diffuse-specular (SDS) light paths between the car interior and the near-specular window. MMLT has a hard time finding the specular light paths, and is often trapped in local modes, producing streaks on the image. MEMLT is able to resolve the static SDS paths more efficiently, but produces ghosting artifacts since it does not move in the time dimension. Our method moves in all dimensions and generates smooth results.

6.1 Limitations

Integrating our method into an existing renderer requires some work, because we need to automatically differentiate the shaders. However, once automatic differentiation has been set up, it is easier to integrate other distributed effects such as motion blur. Automatic differentiation could also be helpful for the shaders/integrators that require the derivatives of the light path (e.g. ray differentials).

As with most MCMC rendering algorithms, high frequency visibility changes can significantly lower the efficiency. Our Gaussian prior reduces this effect but tiny geometry can still cause problems. In addition to visibility changes, there can also be some pathological cases where the path contribution function is extremely noisy. For example, multiple-bounce reflections involving glossy surfaces with high frequency displacement maps. In these cases the derivatives become unreliable, and our method might start to produce correlated noise or have low acceptance rate. We also observe that light transport integration involves both global and local exploration challenges. We need to globally find high-contribution regions, and then locally sample them despite their narrowness. Our method dramatically improves local sampling, but it still needs seed paths that are globally reasonably well distributed. Finally, since the derivatives and covariance computation incurs extra overhead, for relatively simple scenes and BSDFs where ray casting is cheap and isotropic mutation is sufficient, the adaptiveness of our method may not be worth the cost.

7 Conclusions

We presented a novel Hessian-based Hamiltonian Monte Carlo method and applied it to light transport simulation. By introducing Hamiltonian dynamics, we are able to sample from the local quadratic representation that does not define a distribution. Our method can capture the local correlation of the path throughput function, making it suitable for rendering difficult lighting scenarios such as the combination of glossy-to-glossy transport and motion blur. We anticipate that the method's generality will make it possible to render a wider variety of effects such as retroreflective materials, spectral effects, and participating media.

Acknowledgements

We are grateful to the anonymous reviewers for their valuable comments. This work is funded by NSF grants 1451830 and Academy of Finland grant 277833. Wenzel Jakob was supported by an ETH/Marie Curie fellowship.

References

ANDRIEU, C., AND THOMS, J. 2008. A tutorial on adaptive MCMC. *Statistics and Computing 18*, 4, 343–373.

BELCOUR, L., SOLER, C., SUBR, K., HOLZSCHUCH, N., AND DURAND, F. 2013. 5D covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph. 32*, 3, 31.

BELL, B., 2003–2015. CppAD: A package for differentiation of C++ algorithms. `http://www.coin-or.org/CppAD/`.

BETANCOURT, M. 2013. A general metric for riemannian manifold hamiltonian monte carlo. In *GSI 2013*, 327–334.

CHEN, M., AND ARVO, J. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph. 19*, 4, 246–278.

CLINE, D., TALBOT, J., AND EGBERT, P. 2005. Energy redistribution path tracing. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3 (July), 1186–1195.

DUANE, S., KENNEDY, A. D., PENDLETON, B. J., AND ROWETH, D. 1987. Hybrid monte carlo. *Physics Letters B 195*, 2, 216 – 222.

GIROLAMI, M., AND CALDERHEAD, B. 2011. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) (With Discussion) 73*, 123 – 214.

GRIEWANK, A., AND WALTHER, A. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *SCA 2003*.

GRITZ, L., STEIN, C., KULLA, C., AND CONTY, A. 2010. Open shading language. In *SIGGRAPH 2010 Talks*, 33:1–33:1.

GUENTER, B. K. 2007. Efficient symbolic differentiation for graphics applications. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3, 108.

HACHISUKA, T., KAPLANYAN, A. S., AND DACHSBACHER, C. 2014. Multiplexed metropolis light transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4, 100.

HANIKA, J., KAPLANYAN, A., AND DACHSBACHER, C. 2015. Improved half vector space light transport. *Computer Graphics Forum (Proc. EGSR) 34*, 4, 65–74.

HASTINGS, W. K. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*, 1, 97–109.

HOLZSCHUCH, N., AND SILLION, F. X. 1998. An exhaustive error-bounding algorithm for hierarchical radiosity. *Computer Graphics Forum 17*, 4, 197–218.

IGEHY, H. 1999. Tracing ray differentials. *SIGGRAPH 1999*, 179–186.

JAKOB, W., AND MARSCHNER, S. 2012. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4, 58.

JAKOB, W., 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

KAPLANYAN, A. S., HANIKA, J., AND DACHSBACHER, C. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4, 102.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. *Comput. Graph. Forum (Proc. Eurographics) 21*, 3, 531–540.

KETTUNEN, M., MANZI, M., AITTALA, M., LEHTINEN, J., DURAND, F., AND ZWICKER, M. 2015. Gradient-domain path tracing. *ACM Trans. Graph. (Proc. SIGGRAPH) 34*, 4 (July), 123.

KITAOKA, S., KITAMURA, Y., AND KISHINO, F. 2009. Replica exchange light transport. *Computer Graphics Forum 28*, 8 (Dec.), 2330–2342.

LAI, Y., FAN, S., CHENNEY, S., AND DYER, C. 2007. Photorealistic image rendering with population monte carlo energy redistribution. *Rendering Techniques (Proc. EGSR)*, 287–295.

LAI, Y.-C., , LIU, F., AND DYER, C. 2009. Physically-based animation rendering with markov chain monte carlo. *University of Wisconsin - Madison Computer Sciences Department*, UW-CS-TR-1653.

LEAL, J. R., 2011–2015. CppADCodeGen. `https://github.com/joaoleal/CppADCodeGen/`.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., AND AILA, T. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4, 95.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys. 21*, 6, 1087–1092.

NEAL, R. M. 2010. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo 54*, 113–162.

PIPONI, D. 2004. Automatic differentiation, C++ templates, and photogrammetry. *Journal of graphics, GPU, and game tools 9*, 4, 41–55.

RAMAMOORTHI, R., MAHAJAN, D., AND BELHUMEUR, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2.

RITCHIE, D., LIN, S., GOODMAN, N. D., AND HANRAHAN, P. 2015. Generating design suggestions under tight co.traint.ith gradient-based probabilistic programming. *Comput. Graph. Forum (Proc. Eurographics) 34*, 2, 515–526.

ROBERTS, G. O., AND TWEEDIE, R. L. 1996. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli 2*, 4, 341–363.

SCHWARZHAUPT, J., JENSEN, H. W., AND JAROSZ, W. 2012. Practical hessian-based error control for irradiance caching. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6, 193.

SHINYA, M., TAKAHASHI, T., AND NAITO, S. 1987. Principles and applications of pencil tracing. *Comput. Graph. 21*, 4, 45–54.

SUYKENS, F., AND WILLEMS, Y. D. 2001. Path differentials and applications. In *Eurographics Workshop on Rendering Techniques*, 257–268.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for monte carlo rendering. SIGGRAPH 1995, 419–428.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. SIGGRAPH 1997, 65–76.

VEACH, E. 1998. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA.

WALD, I., WOOP, S., BENTHIN, C., JOHNSON, G. S., AND ERNST, M. 2014. Embree: A kernel framework for efficient cpu ray tracing. *ACM Trans. Graph. 33*, 4, 143.

WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Rendering Techniques (Proc. EGSR)*, 195–206.

WARD, G., AND HECKBERT, P. 1992. Irradiance gradients. In *Eurographics Rendering Workshop*, 85–98.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. SIGGRAPH 1988, 85–92.

A Pseudo-code for the algorithm

We show the pseudo-code for our algorithm. Given the gradient G and the Hessian H of the log target function $\log f(x)$, and a user parameter σ^2, our method outputs an anisotropic Gaussian distribution $\Sigma^*, \boldsymbol{\mu}^*$. Note that we simplify the algorithm using the fact that the inverse mass matrix A and H have the same set of eigenvectors.

procedure H2MC(G, H, σ^2) ▷ gradient, Hessian, and prior
  $N \leftarrow$ dimension of the target function
  $T = \frac{\pi}{2}$ ▷ Simulation time
  for $i \leftarrow 1, N$ **do** ▷ Eigendecomposition of H
    $\boldsymbol{e}_i^H \leftarrow i$-th eigenvector of H
    $\lambda_i^H \leftarrow i$-th eigenvalue of H
  end for
  $A = 0_{N \times N}$ ▷ Initialize with zero matrix
  for $i \leftarrow 1, N$ **do** ▷ Construction of A
    $\boldsymbol{e}_i^A \leftarrow \boldsymbol{e}_i^H$
    if $\left|\lambda_i^H\right| > \epsilon$ **then** ▷ ϵ is set to a small number.
      $\lambda_i^A \leftarrow \frac{1}{\left|\lambda_i^H\right|}$
    else
      $\lambda_i^A \leftarrow 0$
    end if
    $A = A + \lambda_i^A \boldsymbol{e}_i^A$
  end for
  for $i \leftarrow 1, N$ **do** ▷ Eigendecomposition of the matrix AH
    $\boldsymbol{e}_i \leftarrow \boldsymbol{e}_i^H$
    if $\left|\lambda_i^H\right| > \epsilon$ **then**
      $\lambda_i \leftarrow \frac{\lambda_i^H}{\left|\lambda_i^H\right|}$ ▷ $\lambda_i^A = \frac{1}{\left|\lambda_i^H\right|}$
    else
      $\lambda_i \leftarrow 0$
    end if
  end for
  $S = 0_{N \times N}$
  $\boldsymbol{o} = 0_{N \times 1}$
  for $i \leftarrow 1, N$ **do** ▷ Scales and offsets (Equation (16))
    $\alpha \leftarrow \lambda_i$ ▷ AH's i-th eigenvalue
    $\beta \leftarrow \lambda_i^A G^T \boldsymbol{e}_i$ ▷ Projection of AG on $\boldsymbol{e}_i$
    if $\lambda_i > 0$ **then**
      $s_i \leftarrow \frac{\exp\left(\sqrt{\alpha}T\right) - \exp\left(-\sqrt{\alpha}T\right)}{2\sqrt{\alpha}}$
      $o_i \leftarrow \frac{\beta}{2\alpha}\left(\exp\left(\sqrt{\alpha}T\right) + \exp\left(-\sqrt{\alpha}T\right) - 1\right)$
    else if $\lambda_i < 0$ **then**
      $s_i \leftarrow \frac{1}{\sqrt{-\alpha}} \sin\left(\sqrt{-\alpha}T\right)$
      $o_i \leftarrow \frac{\beta}{\alpha}\left(\cos\left(\sqrt{-\alpha}T\right) - 1\right)$
    else
      $s_i \leftarrow T$
      $o_i \leftarrow -\frac{\beta T^2}{2}$
    end if
    $S = S + s_i \boldsymbol{e}_i$ ▷ Equation (18)
    $\boldsymbol{o} = \boldsymbol{o} + o_i \boldsymbol{e}_i$
  end for
  $\Sigma = SAS^T$ ▷ Equation (19)
  $\boldsymbol{\mu} = \boldsymbol{o}$
  $\Sigma^* = \left(\Sigma^{-1} + \frac{1}{\sigma^2}\right)^{-1}$ ▷ Prior multiplication (Equation (20))
  $\boldsymbol{\mu}^* = \Sigma' \Sigma \boldsymbol{\mu}$
  return $\Sigma^*, \boldsymbol{\mu}^*$
end procedure