Geometry-Aware Metropolis Light Transport

HISANARI OTSU, Karlsruhe Institute of Technology and The University of Tokyo
JOHANNES HANIKA, Karlsruhe Institute of Technology
TOSHIYA HACHISUKA, The University of Tokyo
CARSTEN DACHSBACHER, Karlsruhe Institute of Technology

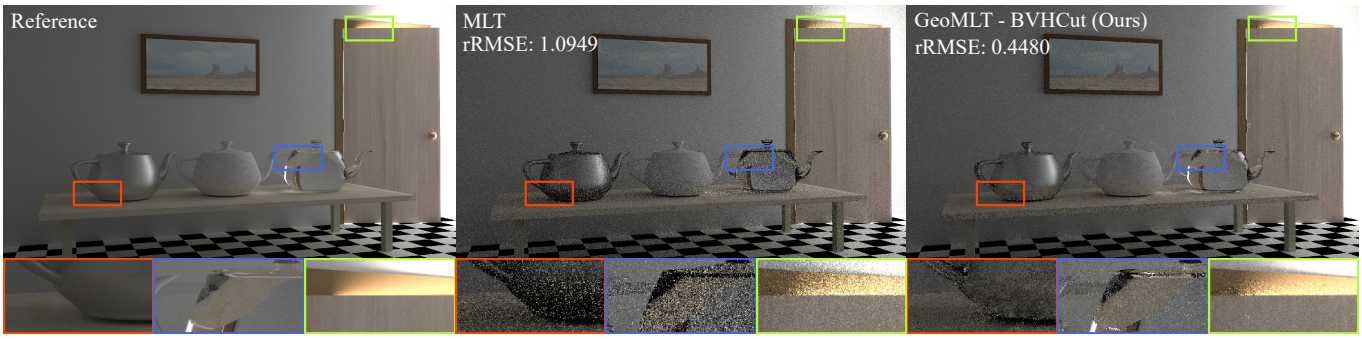

Fig. 1. Equal-time comparison (30 minutes) of the *Ajar door* scene, which is only illuminated by light leaking through the door. Left: Reference image computed with bidirectional path tracing. Middle: Image rendered with Metropolis light transport (MLT) [Veach and Guibas 1997]. Right: Image rendered with our *geometry-aware Metropolis light transport* (GeoMLT). Note how MLT has difficulties as the mutation of the paths passing through the gap tends to be rejected because of changing visibility. Our approach adaptively controls the mutation size according to the geometry information surrounding each path segment.

Markov chain Monte Carlo (MCMC) rendering utilizes a sequence of correlated path samples which is obtained by iteratively mutating the current state to the next. The efficiency of MCMC rendering depends on how well the mutation strategy is designed to adapt to the local structure of the state space. We present a novel MCMC rendering method that automatically adapts the step sizes of the mutations to the *geometry* of the rendered scene. Our geometry-aware path space perturbation largely avoids tentative samples with zero contribution due to occlusion. Our method limits the mutation step size by estimating the maximum opening angle of a cone, centered around a segment of a light transport path, where no geometry obstructs visibility. This geometry-aware mutation increases the acceptance rates, while not degrading the sampling quality. As this cone estimation introduces a considerable overhead if done naively, to make our approach efficient, we discuss and analyze fast approximate methods for cone angle estimation which utilize the acceleration structure already present for the ray-geometry intersection. Our new approach, integrated into the framework of Metropolis light transport, can achieve results with lower error and less artifact in equal time compared to current path space mutation techniques.

CCS Concepts: • **Computing methodologies** → **Ray tracing**;

Additional Key Words and Phrases: Markov chain Monte Carlo light transport, global illumination

ACM Reference Format:
Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. 2018. Geometry-Aware Metropolis Light Transport. *ACM Trans. Graph.* 37, 6, Article 278 (November 2018), 11 pages. https://doi.org/10.1145/3272127.3275106

1 INTRODUCTION

Light transport simulation with Markov chain Monte Carlo (MCMC) is attracting increasing attention in the rendering research community due to its excellent ability to locally explore features in the sampling domain. MCMC employs a Markov chain whose states are path samples distributed according to a user-defined target distribution. Thanks to this characteristic, MCMC rendering can efficiently focus on sampling transport paths with large contributions to the pixels in the image.

MCMC rendering proceeds by generating a tentative sample based on the current sample which may then be accepted as the next state, for instance by using a Metropolis-Hastings acceptance probability. The efficiency of MCMC rendering crucially depends on the design of the transition kernel (path mutation): very small steps tend to be accepted often as the samples are likely to be very similar and the acceptance probability will be high. The efficiency of MCMC rendering, however, also depends on the autocorrelation between states. High autocorrelation means less additional information in the samples and thus lower effective sample size. Large step sizes, on the other hand, typically result in many rejected samples and can cause chains getting stuck. In order to design an efficient kernel, we thus need to exploit the characteristics of the local structure of the target distribution.

We focus on inefficiency due to rejections of samples caused by *geometric visibility*, which has not been addressed so far by existing approaches. For instance, let us think about a light transport path passing through a small gap (Fig. 2). The mutation on the first part of the path involves a change of the outgoing direction of the segment. Here, the kernel with the support of (b) is better than that of (a)

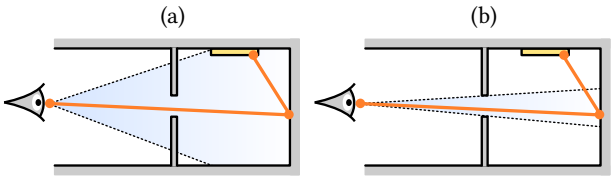

Fig. 2. Path mutations can lead to many rejected proposals due to changes in visibility: In this example, we mutate a light transport path (orange) by changing the outgoing direction at the vertex on the sensor; the blue regions show possible supports of the mutation kernels. Mutation by the kernel with bigger support (a) is more likely to lead to an occluded segment and thus rejected than mutations with kernel (b).

since much of the support of (a) results in tentative paths with zero contribution due to the change in visibility. On the other hand, the smaller kernel is not always better because it can degrade the exploration and result in higher autocorrelation. Existing adaptive mutation approaches [Jakob and Marschner 2012; Li et al. 2015] do not care about the change of visibility in the design of their mutations, and thus suffer from inefficiencies due to potentially generating invisible paths.

We leverage the *geometry* information of the scene for a mutation technique for MCMC rendering for the first time. Our method controls mutation by estimating a cone centered around a path segment where no scene geometry would obstruct a perturbed segment. To make our algorithm efficient, we introduce an approximate cone angle estimation taking advantage of the acceleration structure already present for ray tracing. Our mutation strategy can be easily integrated into existing path space mutation techniques such as Metropolis light transport [Veach and Guibas 1997], providing a tool to take geometry information into account for MCMC rendering. In summary, the main contributions of our work are:

- An MCMC rendering algorithm which exploits geometric information and takes visibility into account to better adapt to the local structure of the target distribution.
- A fast approximate estimation of the maximum cone angle centered around a segment of a light transport path such that no scene surfaces are contained in the cone.
- A practical implementation of a geometry-aware mutation strategy for Metropolis light transport, which only requires simple modifications to existing code.

2 BACKGROUND AND PREVIOUS WORK

2.1 Light Transport Simulation

Light transport simulation computes a solution of the *path integral*, which determines the intensity I_j of the j-th pixel as an integral over the measurement contribution function $f_j(\bar{x})$ with respect to the product area measure μ:

$$I_j = \int_\Omega f_j(\bar{x}) d\mu(\bar{x}), \tag{1}$$

where Ω denotes the path space which represents the space of all possible paths of arbitrary length. Specifically, the path space can be written as $\Omega = \cup_{k=2}^{\infty} \Omega_k$, where Ω_k is the set of paths with k vertices. An element of the path space $\bar{x} \in \Omega$ is called a *path* and

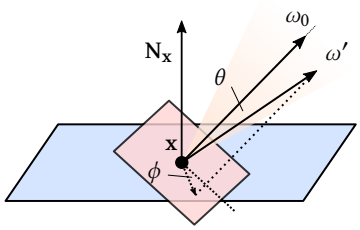

Fig. 3. Perturbation of the outgoing direction at a vertex (not located on the sensor). Instead of using the tangent plane at $\mathbf{x}$ (blue plane), we perturb the original direction ω to the direction ω' in the local frame with the up vector ω (red plane). In this local frame, the perturbed direction can be represented by the spherical coordinates (θ, ϕ) where θ is exponentially distributed and ϕ is sampled uniformly. The kernel support (orange) is controlled by the two parameters r_{min} and r_{max}.

represented by a sequence of points on the scene surfaces $\mathcal{M}$ as $\bar{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \Omega_k$ where $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathcal{M}$. In the following discussion we will drop the subscript j from the contribution function f_j for notational simplicity.

Many techniques to sample paths $\bar{x}$ have been proposed. Path tracing [Kajiya 1986], for example, generates paths starting from the sensor, light tracing [Arvo 1986] traces paths from the light sources, and bidirectional path tracing [Lafortune and Willems 1993; Veach and Guibas 1994] generates paths starting from the both sides.

2.2 MCMC Rendering

MCMC has initially been introduced to rendering as Metropolis light transport (MLT) [Veach and Guibas 1997]. This method generates a sequence of samples in the path space based on the Metropolis-Hastings (MH) algorithm [Hastings 1970; Metropolis et al. 1953]. New samples are generated based on the previous samples only, which results in a Markov chain. When taking the measurement contribution function f as target distribution, the sequence of paths is eventually distributed according to the normalized measurement contribution function f/b, where $b = \int_\Omega f(\bar{x}) d\mu(\bar{x})$ is the normalization constant. The value b is typically estimated with independent MC sampling, e.g., bidirectional path tracing.

Given the current path $\bar{x}_i$, the MH algorithm proposes a tentative path $\bar{y} \sim T(\bar{x} \to \bar{y})$ according to the transition kernel T. This proposed tentative path $\bar{y}$ is either rejected (and the current state kept) or accepted as the next state with the *acceptance probability* $\min(1, a)$:

$$\bar{x}_{i+1} = \begin{cases} \bar{y} & \text{with probability } \min(1, a) \\ \bar{x}_i & \text{otherwise,} \end{cases} \tag{2}$$

where

$$a = a(\bar{x}_i \to \bar{y}) = \frac{f^*(\bar{y}) T(\bar{y} \to \bar{x}_i)}{f^*(\bar{x}_i) T(\bar{x}_i \to \bar{y})} \equiv \frac{R(\bar{x}_i \to \bar{y})}{R(\bar{y} \to \bar{x}_i)}. \tag{3}$$

Here f^* is the scalar contribution function, which typically is the luminance of f. The equation is further simplified with $R(\bar{x}_i \to \bar{y}) = f(\bar{y})/T(\bar{x}_i \to \bar{y})$. Using the sequence of samples $\{\mathbf{x}_i\}_{1,\ldots,N}$ generated by this mutation process, the estimate of the pixel intensity I_j

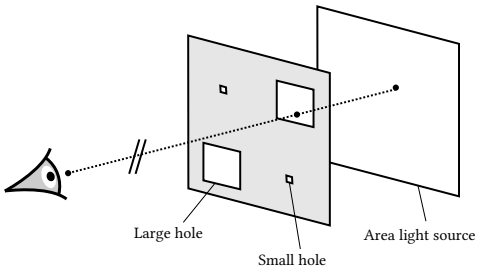

Fig. 4. The scene configuration of our introductory example. The scene is lit by a large area light, occluded from the camera by a plane with two small and two large quadratic holes. This example illustrates the conceptual difference between standard and geometry-aware MLT (see also Fig. 5).

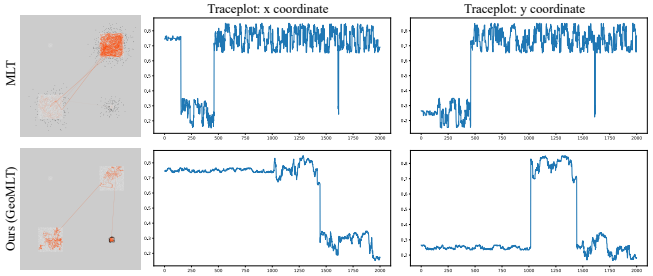

Fig. 5. The trace-plot of the samples for the scene configuration shown in Fig. 4 for MLT and our GeoMLT. Left: the trace-plots in screen space. In this example the state is a single path segment with two vertices. The screen position is calculated from the direction of the segment and projection onto the screen. The orange lines show the exploration of the Markov chains. The black dots denote rejected samples; rejection in this example is due to change of visibility only. Right: the trace-plots of the x-coordinates of screen-projection for the first 2000 mutations.

can be written as

$$I_j \approx \hat{I}_j \equiv \frac{1}{N}\sum_{i=1}^{N}\frac{f(\bar{x}_i)}{f^*(\bar{x}_i)/b} = \frac{b}{N}\sum_{i=1}^{N}\frac{f(\bar{x}_i)}{f^*(\bar{x}_i)}. \tag{4}$$

Mutation strategies depend on a state space. Kelemen et al. [2002] proposed primary sample space MLT (PSSMLT), a variant of MLT where the state is defined as a sequence of (random) numbers as opposed to a sequence of path vertices. Extending this framework, Hachisuka et al. [2014] proposed multiplexed MLT (MMLT) which combines PSSMLT with multiple importance sampling [Veach and Guibas 1995] by extending the state space to additionally pinpoint the bidirectional path connection strategy. MLT operates in the path space and cannot directly be combined with PSSMLT. Three recent works by Pantaleoni [2017], Otsu et al. [2017], and Bitterli et al. [2017] concurrently proposed techniques to combine the different state spaces by using an inverse mapping from the primary sample space to the path space.

2.3 Path Space Perturbations

Our geometry-aware mutation technique is based on the *lens*, *caustic*, and *multi-chain* perturbations operating in the path space [Veach and Guibas 1997]. Here a subpath is perturbed by changing the outgoing direction from a vertex in spherical coordinates [Veach and Guibas 1997, Sec. 5.3.2]. For instance, the lens perturbation changes the direction $\mathbf{x}_1 \rightarrow \mathbf{x}_2$ and traces specular interactions until it finds a diffuse surface.

Fig. 3 illustrates such a perturbation which is performed on the unit sphere (θ, ϕ) in the local frame. Veach and Guibas [1997] suggested to perturb the direction in the local frame by the reciprocal distribution as

$$\theta = r_{\max}\exp\left(-\ln\frac{r_{\min}}{r_{\max}}\cdot U_1\right), \quad \phi = 2\pi\cdot U_2, \tag{5}$$

where $r_{\min} < r_{\max}$ are the parameters controlling the kernel size such that $\theta \in [r_{\min}, r_{\max}]$; U_1 and U_2 are uniform random numbers in $[0, 1]$. The probability density function (pdf) in the solid angle measure $d\sigma$ is then

$$p_\sigma(\omega_0 \rightarrow \omega(\theta,\phi)) = \begin{cases} \frac{1}{\pi\theta\sin\theta} & \text{if } \theta \in [r_{\min}, r_{\max}] \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The step size has a large influence on the rendering performance: small steps lead to high autocorrelation and a low number of effective samples, and large steps result in many rejections and might lead to stuck chains. Finding a good step size thus is crucial for the overall performance; our work adaptively controls this step size based on the geometry surrounding a path.

2.4 Adaptive Step Sizes

Several researchers proposed to adaptively change the step size according to certain features of the measurement contribution function. Li et al. [2015] utilized local information obtained from analytic derivatives to adaptively control the shape of the transition kernels based the idea of Hamiltonian Monte Carlo [Duane et al. 1987]. Jakob and Marschner [2012] introduced a mutation strategy which explores the state space constrained to the lower dimensional subspace of valid transport paths on specular surfaces. Kaplanyan et al. [2014] and Hanika et al. [2015] generalized this approach by introducing a representation of paths based on a sequence of the half-vectors. This space of half vectors is explored using adaptive step sizes in all dimensions derived from ray differentials. These approaches are all based on analytic derivatives and ignore visibility since visibility derivatives contain Dirac delta functions, and consequently can suffer from inefficiency due to sampling obstructed paths. Our work provides a first tool to incorporate visibility into mutation strategies of MCMC rendering.

2.5 Cones in Rendering

Cones have been used in various contexts in rendering. Cone tracing [Amanatides 1984] extended a ray to a cone as a primitive for scene intersection to achieve efficient anti-aliasing. Roger et al. [2007] used cones to represent collections of rays and constructed a ray-hierarchy to determine ray-scene intersections for multiple coherent rays. Mora [2011] utilized cones as ray packets in the context of divide-and-conquer ray tracing. Crassin et al. [2011] used cone tracing for voxelized scene geometries. These approaches all essentially find an intersection between a given cone and the scene

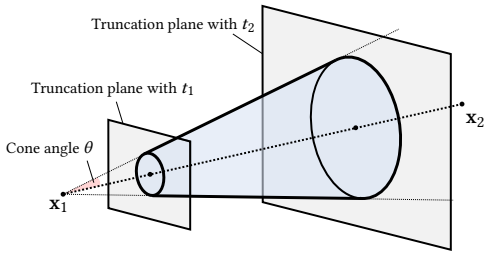

Fig. 6. Parametrization of the truncated, isotropic cone for a path segment $\mathbf{x}_1\mathbf{x}_2$. θ is the cone angle which is defined as the angle between the cone axis vector $\mathbf{x}_2 - \mathbf{x}_1$ and the surface of the cone. The truncation range $[t_1, t_2]$ describes the valid domain of the cone which lies in the slab defined by two planes perpendicuar to the cone axis. We note that the truncation plane is not always perpendicular to the cone axis.

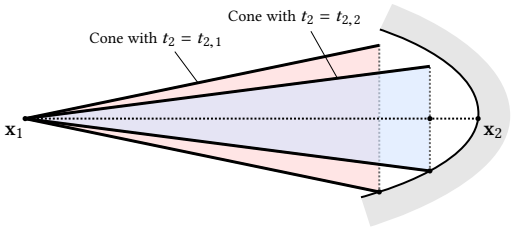

Fig. 7. Intersection between the cone and the object at $\mathbf{x}_2$. We utilize the parameter t_2 to control how much close-by geometry in the target region influences the cone angle. The two cones with different parameters $t_{2,1}$ and $t_{2,2}$ ($t_{2,1} < t_{2,2}$) have different opening angles, because the base of the cones intersect geometry close to $\mathbf{x}_2$.

geometry. In contrast, our approach fits empty cones around a given ray segment to determine empty space.

3 OVERVIEW

The main idea of our geometry-aware MLT is to mutate the direction of a path segment such that it will not result in occlusion due to the scene geometry. To design such a mutation, we find a cone centered around a path segment and determine the maximum opening angle such that no geometry is intersected with this cone (Sec. 4). This operation will be performed very often and thus has to be efficient to achieve a net performance gain. We exploit the fact that, for MCMC, this cone angle estimation does not need to be exact. We thus present approximations which exploit the very same data structure as already used for the ray-triangle intersection (Sec. 5). We then explain how to incorporate our geometry-aware mutation into Metropolis light transport (Sec. 6).

Fig. 4 and 5 illustrate and motivate our approach. The shown scene consists of a blocker with four holes and is illuminated by an area light source in the back. The surface between the sensor and the light source occlude the light and leave two smaller and two larger holes. In this example, we restricted paths to a length of one (i.e. the light source is directly visible from the sensor or blocked) to illustrate the conceptual difference of conventional and geometry-aware MLT. We combined two types of mutations for this example: the lens perturbation which is designed to mutate within a hole, and the bidirectional mutation for jumps between the holes. We ran the simulation for both MLT and geometry-aware MLT (Fig. 5). The mutation size for standard MLT is configured to explore the larger hole well. The perturbation with the same mutation size, however, cannot explore the small hole very well as most proposals have zero contribution due to occlusion. On the other hand, our approach can explore all the holes well as it adapts the mutation size to the surrounding geometry around the path segment. This can also be observed in the trace-plot of the x-coordinates of the projected states. The top-right plot shows the case with naive MLT where the mutation explores the larger hole well, yet often fails to sample the smaller hole. On the other hand, the plot for geometry-aware mutations shows good exploration of both holes by adapting the mutation size.

4 GEOMETRY-AWARE PERTURBATION SIZE

Our mutation technique avoids rejections due to a change in visibility by defining the maximum perturbation size $r_{\max}$ (Eq. 5) as a function of the current path $\bar{x}$. Depending on $r_{\max}$, we choose the minimum perturbation size as $r_{\min}(\bar{x}) = \alpha r_{\max}(\bar{x})$ with a user-defined parameter $\alpha \in [0, 1)$.

We first introduce the idea with a single path segment $\mathbf{x}_1\mathbf{x}_2$ where $\mathbf{x}_1$ is kept and the outgoing direction is to be perturbed. We directly apply this in the lens perturbation and potentially iterate for more segments along a path when doing multichain perturbations. The maximum kernel size of the perturbation is related to the set of segments which do not intersect any other surface than that where $\mathbf{x}_2$ lies.

To this end, we determine a truncated cone (Fig. 6) where $\mathbf{x}_1$ is the apex and $\mathbf{x}_2$ is on the axis of the cone with the maximum cone angle θ (with $0 < \theta < \pi/2$) such that no other surface intersects with it. The truncation of the cone's apex and the base is used to avoid self-intersections close to $\mathbf{x}_1$ and to avoid unnecessary small cone angles due to geometry close to $\mathbf{x}_2$ (Fig. 7). These surfaces do not necessarily represent blocking geometry and we thus truncate the cone. We denote the truncation range as $[t_1, t_2]$ measured along the cone axis from the point $\mathbf{x}_1$. Furthermore, we allow the cone base at t_2 to be non-perpendicular to the cone axis $\mathbf{x}_1\mathbf{x}_2$ to better account for oblique directions and therefore take the geometric normal of the surface intersected at $\mathbf{x}_2$ as its normal. The base at t_1 is kept perpendicular to the cone axis.

We denote the set of all points contained in this truncated cone as $C(\theta)$ (we omit the dependency on $\mathbf{x}_1$ and $\mathbf{x}_2$ for notational simplicity), then the set of points contained both in the truncated cone and the scene surface $\mathcal{M}$ can be written as $\mathcal{M} \cap C(\theta)$. Determining the kernel size thus requires:

$$\theta_{\max} = \sup\{\theta \mid \mathcal{M} \cap C(\theta) = \varnothing\}. \tag{7}$$

Unfortunately, determining $\theta_{\max}$ is very costly. One possibility would be a binary search with a cone-scene intersection test. The cone angle, however, is required for every perturbation of a path segment and an exact computation is prohibitively expensive – we thus need efficient approximations.

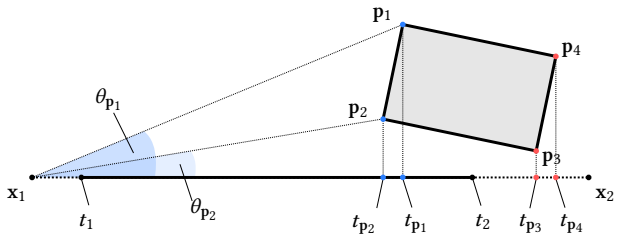

Fig. 8. Calculation of the approximate maximum cone angle for geometry enclosed by an AABB b with vertices $V(b) = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ (illustrated in 2D). The projected distances of $\mathbf{p}_1$ and $\mathbf{p}_2$ (blue dots) are within the range of $[t_1, t_2]$ and we use these vertices to estimate the cone angle; the projected distances of $\mathbf{p}_3$ and $\mathbf{p}_4$ (red dots) are outside of this interval and ignored.

5 APPROXIMATE CONE ANGLE

To alleviate the computational cost for determining $\theta_{\max}$, we now introduce approximations whose computation leverages the same acceleration structure as already used for ray tracing. We will use the axis-aligned bounding boxes (AABBs) of the scene geometry as proxy geometries for the estimation of $\theta_{\max}$. Note that the perturbation size does not need to be exact as long as it preserves the detailed balance property of the Markov chain.

5.1 Estimating the Cone Angle for a Single AABB

The first building block of the estimation is computing the cone angle $\theta_{\max}$ for a given segment $\mathbf{x}_1\mathbf{x}_2$ and a single AABB; we here assume that t_1 and t_2 have already been determined to define the truncation plane at t_1 (perpendicular to $\mathbf{x}_1\mathbf{x}_2$) and at t_2 whose orientation is defined by the geometric normal at $\mathbf{x}_2$. We denote the AABB as b and its corners as $\mathbf{p_i}$, and compute $\theta_{\max}$ for the *truncated* cone. For this, we take the minimum of the angles computed in following three stages (Fig. 8):

a) if $\mathbf{p_i}$ lies in between the two truncation planes, we consider it for $\theta_{\max}$ and compute the angle between $\mathbf{x}_1\mathbf{x}_2$ and $\mathbf{x}_1\mathbf{p_i}$.

b) we compute the closest point to $\mathbf{x}_1\mathbf{x}_2$ on each edge of the AABB. If the point is between the two truncation planes, we compute the respective angle and update $\theta_{\max}$.

c) lastly we test if the base contains the closest point on the AABB. We intersect the truncation plane at t_2 with the AABB. This yields a convex polygon and computing the closest point to $\mathbf{x}_1\mathbf{x}_2$ reduces to a 2D problem.

5.2 Cone Estimation using a BVH-Cut

To estimate the maximum cone angle for a segment and the *entire scene*, we reuse the ray tracing acceleration structure. The AABBs that we consider for the approximation form a cut through the tree of the acceleration structure. We begin by computing an initial cut which will be refined afterwards. It contains all nodes whose parents are intersected by $\mathbf{x}_1\mathbf{x}_2$, but which are not intersected themselves (Fig. 10). This requires only minor modifications to common ray traversal code (Alg. 1). This initial cut contains AABBs which surround the segment $\mathbf{x}_1\mathbf{x}_2$ but do not intersect it. This also avoids the case that the cone angle becomes zero if the path goes through the empty space of an AABB. In this case, the ray segment is already intersected with the AABB and thus it would be excluded from C_{init} and the estimation of cone angles.

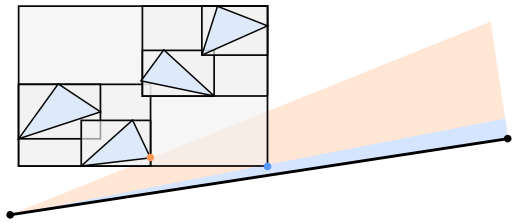

Fig. 9. Using the AABBs of the geometry always underestimates the cone angle. The refinement yields more accurate results.

Computing $\theta_{\max}$ from this set of AABBs in C_{init} using the algorithm from Sec. 5.1 yields estimated cone angles which will always be smaller than the accurate solution (Fig. 9).We improve the accuracy by refining the cut, i.e., replacing nodes in the cut by their children. We iteratively split the node which currently limits the cone angle. If this node is a leaf, we terminate the refinement; otherwise we proceed until a maximum number $N_{\max}$ of AABBs is reached. Alg. 2 shows an efficient refinement algorithm using a priority queue.

5.3 Trading Accuracy for Speed

We also explored variants of cone angle estimations which are faster to evaluate but more approximate. In particular we show results for the variant which implements option a) from Sect. 5.1, i.e., testing only AABB corners, and without cut refinement. In Sect. 7 we evaluate this more approximate and faster method as well as the more precise method to cover the spectrum of possibilities. In general, it often paid off to use the more accurate cone estimation.

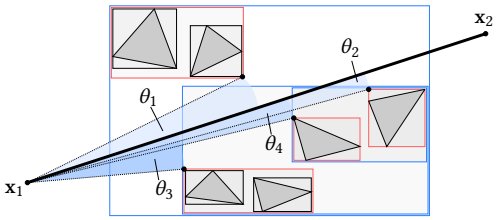

Fig. 10. Computing the approximate maximum cone angle for an entire scene using the AABBs of a bounding volume hierarchy. Blue rectangles show the AABBs intersected by the truncated ray, and the red rectangles show their children which are not intersected themselves. These are used for estimating the cone angles.

6 GEOMETRY-AWARE METROPOLIS LIGHT TRANSPORT

We will now introduce *geometry-aware Metropolis light transport* which uses the cone angle estimation for adapting the step sizes. We begin with a geometry-aware extension of the multi-chain perturbation. We can similarly extend the other perturbation techniques, such as the lens or caustic perturbation. As the original multi-chain

ALGORITHM 1: A standard ray traversal procedure modified to calculate the initial set of AABBs C_{init}. The input is the truncated ray r and the current node v. The function begins the traversal from the root node and C_{init} is initialized with $\varnothing$. The blue line shows the modification to a regular intersection computation.

```
Function IntersectWithInitialCut(r, v):
    if ray r does not intersect AABB of v then
        C_init ← C_init ∪ {v}
    else
        if child(v) = ∅ then
            IntersectWithPrimitives(v)
        else
            foreach v_c ∈ child(v) do
                IntersectWithInitialCut(r, v_c)
            end
        end
    end
EndFunction
```

ALGORITHM 2: Greedy refinement process of BVH-cut. r is the ray segment and $N_{\max}$ is the maximum number of AABBs considered for the approximation. C_{init} is the initial cut computed during ray traversal. Q is a priority queue and PopQueue returns the node v among the elements in the queue which limits the cone angle.

```
Function RefineBVHCut(r, N_max):
    Q ← ∅
    foreach v ∈ C_init do
        PushQueue(Q, v, θ computed for AABB v)
    end
    while number of elements in Q < N_max do
        v ← PopQueue(Q)
        if child(v) = ∅ then
            return v
        else
            foreach v_c ∈ child(v) do
                PushQueue(Q, v_c, θ computed for AABB v)
            end
        end
    end
    return PopQueue(Q)
EndFunction
```

mutation, given a current path $\bar{x} = \mathbf{x}_1\mathbf{x}_2\cdots\mathbf{x}_k$, our modified strategy iteratively mutates the path segments $\mathbf{x}_i\mathbf{x}_{i+1}$ starting off a non-specular surface and ending on a specular surface until it finds two consecutive non-specular vertices $\mathbf{x}_{s-1}\mathbf{x}_s$; the mutation examines segments starting from the point on the aperture ($\mathbf{x}_k$). Note that this mutation retains the number of vertices and the surface type associated with each vertex, and the mutated subpath then is $\bar{y} = \mathbf{x}_1\mathbf{x}_2\cdots\mathbf{x}_{s-1}\mathbf{y}_s\cdots\mathbf{y}_k$. The perturbation of each segment $\mathbf{x}_i\mathbf{x}_{i+1}$ ($k > i \geq s$) is done as described in Sec. 2.3 where $r_{\max}$ is the estimated maximum cone angle, and $r_{\min}(\mathbf{x}_i\mathbf{x}_{i+1}) = \alpha r_{\max}$ with a constant parameter $\alpha \in (0, 1)$.

If $\mathbf{x}_{i+1}$ is on the aperture or a non-specular surface, we perturb the original direction $\omega_0 = \mathbf{x}_{i+1} \to \mathbf{x}_i$. The next vertex $\mathbf{y}_i$ is obtained by tracing a ray into the mutated direction ω. The direction $\mathbf{y}_{i+1} \to \mathbf{x}_i$ is not used because it would introduce an additional occlusion test between $\mathbf{y}_{i+1}$ and $\mathbf{x}_i$ when we compute the acceptance ratio. If the vertex $\mathbf{x}_{i+1}$ is on a specular surface, we trace a ray into the deterministic direction to find the next vertex $\mathbf{y}_i$.

Scene	Method	Mutations	Acceptance rate	
			Overall	Perturbation
Ajar door	MLT	490M	15.3%	20.0%
	FastApprox	386M	34.1%	57.6%
	BVHCut	242M	35.2%	59.8%
Dining room	MLT	570M	8.7%	11.3%
	FastApprox	414M	27.7%	49.2%
	BVHCut	256M	31.2%	56.3%
Salle de bain	MLT	557M	22.0%	30.8%
	FastApprox	466M	36.3%	59.4%
	BVHCut	292M	39.7%	66.2%
Staircase	MLT	359M	20.3%	30.3%
	FastApprox	246M	32.5%	54.7%
	BVHCut	134M	39.9%	69.4%

Table 1. Statistics of our experiments. Mutations shows how many mutations is executed in the same rendering time. We also show that the overall acceptance rates and the rates for the corresponding perturbation technique to the method.

The mutated path $\bar{y}$ is accepted as the next state with the acceptance probability given in Eq. 3; otherwise the state remains $\bar{x}$. $R(\bar{x} \to \bar{y})$ and $R(\bar{y} \to \bar{x})$ are required to compute the probability. Instead of directly computing them, we can first extract the common term in both and only compute the rest for the subpaths $\bar{x}_s$ and $\bar{y}_s$:

$$R(\bar{x} \to \bar{y}) = C(\bar{x} \leftrightarrow \bar{y}) \cdot \frac{f(\bar{y}_s)}{T(\bar{x}_s \to \bar{y}_s)}, \tag{8}$$

where the shared term $C(\bar{x} \leftrightarrow \bar{y}) \equiv C(\bar{y} \leftrightarrow \bar{x})$ is eventually canceled out when we compute $R(\bar{x} \to \bar{y})/R(\bar{y} \to \bar{x})$. $f(\bar{y}_s)$ and $T(\bar{x}_s \to \bar{y}_s)$ are defined for the subpaths $\bar{x}_s$ and $\bar{y}_s$ as

$$f(\bar{y}_s) = W(\mathbf{x}_k) \cdot \prod_{i=s}^{k-1} f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \cdot G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \tag{9}$$

$$T(\bar{x}_s \to \bar{y}_s) = \prod_{i=k}^{s-1} \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is specular,} \\ p_\sigma(\omega_i' \to \omega_i)\left|\frac{d\sigma}{dA}\right| & \text{otherwise.} \end{cases} \tag{10}$$

We assume in this notation that $f_s(., \mathbf{x}, .)$ evaluates to $L_e(\mathbf{x})$ if $\mathbf{x}$ is on the light source, and further that the BSDF of a specular dielectric contains only the Fresnel term, i.e. the Dirac delta is canceled out in both the BSDF and the transition probability.

We need the Jacobian determinant $|d\sigma/dA|$ to convert the transition probability in Eq. 6 to the area measure, which is a standard geometry term. The estimation of the maximal cone angle is required when we evaluate the term $p_\sigma(\omega_i' \to \omega_i)$. As we compute R for both directions, we also need to compute the maximum cone angle for the reverse probability, which incurs additional computation cost. This cost is not overly expensive because the limited truncation range allows early termination of BVH traversal.

We always combine geometric-aware mutations with bidirectional mutations which enables global exploration of the state space. The sample distribution via MLT converges to the target distribution as long as the *combination* of mutation techniques preserves ergodicity. This is an inherent property from MLT, irrespective of the use of our geometric mutations.

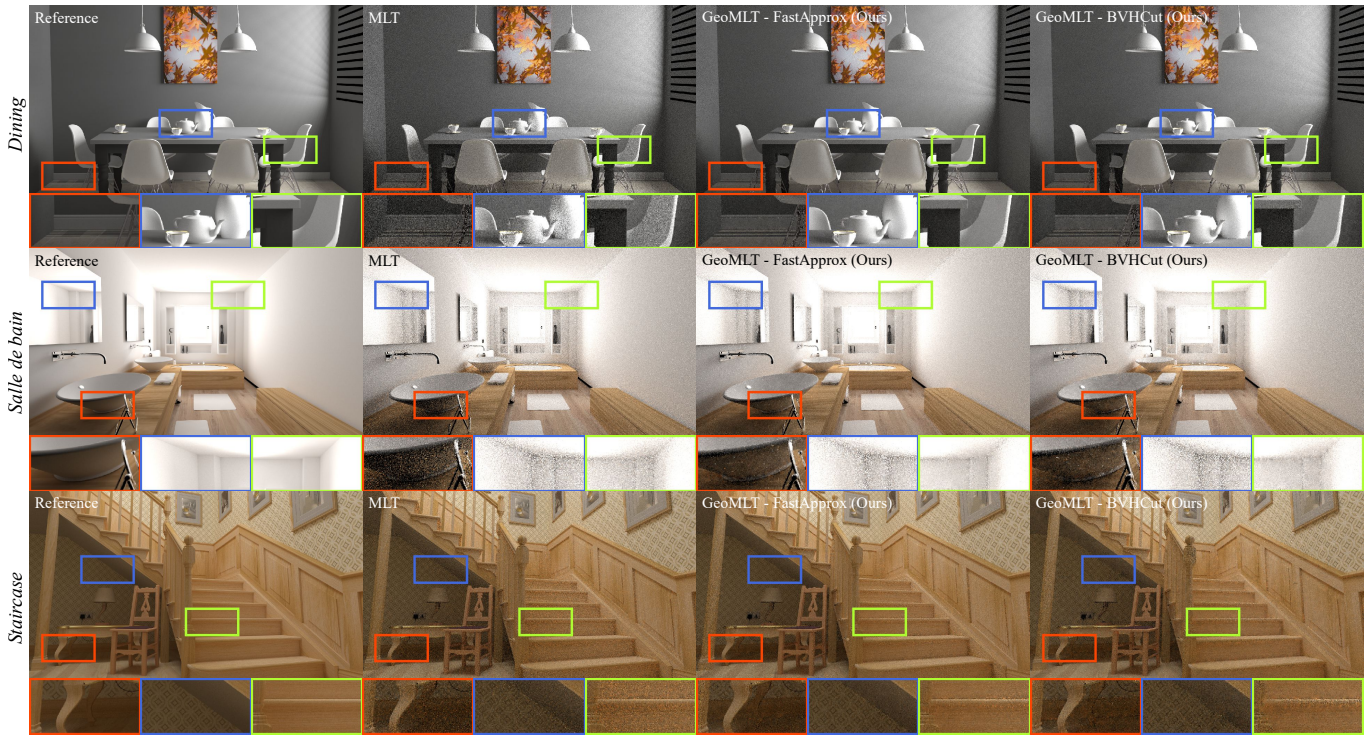

Fig. 11. Equal-time comparisons of the scenes (*Dining room*, *Salle de bain*, and *Staircase*) between MLT and GeoMLT (Ours).

7 RESULTS

We implemented the original and our proposed perturbation techniques in the same rendering system using MLT [Veach and Guibas 1997]. We always refer to the original approach *MLT* and ours as *GeoMLT*. For our method, we implemented geometry-aware multi-chain perturbation. All experiments are conducted on a machine with an Intel Core i7-6700 at 3.4 GHz using 8 threads. All comparisons are equal-time renderings of 30 minutes. The reference images have been rendered using bidirectional path tracing with more than 10 hours of computation time using two machines with 128 and 160 threads. We configured the maximum path length to 14 for all the scenes. We set $r_{\min} = 0.05$ and $r_{\max} = 0.5$ for the mutation techniques without geometry-aware mutations. Furthermore, for our approach, we set $N_{\max} = 10$ and the parameter α is fixed to $\alpha = r_{\min}/r_{\max} = 0.1$ so that the ratio of the two parameters remains the same as for MLT.

For all the experiments, we combined two different classes of mutation techniques: the bidirectional mutation for global exploration, and the multi-chain perturbations for local exploration with equal selection probability. The comparisons are designed to evaluate the effectiveness of our approach for the otherwise difficult cases, and in order to examine how our approach is different from the existing one. For this we use two mutation techniques that differ only in whether they use geometry-aware mutations

Clamping the Estimated Cone Angle. In order to avoid the case that the cone angle degenerates to zero if a ray segment glances off the geometry and the case that no intersection results in overestimation of the cone angle, we clamp the estimated angle to a user-defined range of $[\tau_{min}, \tau_{max}]$. For the experiments in this paper, we set $\tau_{\min} = 0.01$ and $\tau_{\max} = 0.5$. We intentionally set $\tau_{\max}$ to match $r_{\max}$ so that the maximum possible cone angle becomes the same in the comparisons.

Avoiding Unnecessary Cone Angle Underestimation. As shown in Fig. 7, the cone angle will be underestimated if the base of the cone intersects geometry near the target point $\mathbf{x}_2$. In our implementation, we set $t_1 = 10^{-4}$ and $t_2 = 0.99L$ where $L = \|\mathbf{x}_2 - \mathbf{x}_1\|$ is the length of the segment $\mathbf{x}_1\mathbf{x}_2$. This resembles the practice of limiting the valid range of the ray to avoid self-intersections during ray-triangle intersection.

Equal-Time Comparisons. We rendered three scenes with different geometric complexity with different materials: *Ajar door*, *Salle de bain*, and *Staircase*. *BVHCut* refers to the cone angle estimation described in Sec. 5.2, and *FastApprox* refers to Sec. 5.3. Table 1 summarizes the statistics of the renderings for each scene.

Fig. 1 shows an equal-time comparison between MLT and GeoMLT in the *Ajar door* scene. In this scene the geometry seen from the sensor is illuminated by light leaking through the door only. Mutating paths connected through the gap is difficult because of occlusion. GeoMLT can handle these cases efficiently because the mutation step sizes are adapted to the surrounding geometry. GeoMLT well also explores paths near the discontinuities between objects, which improves the efficiency of the MCMC process, for instance, near the boundary between the wall and the floor, or between the teapot and the table. In parts of the scene where paths are unobstructed and are free to scatter diffusely the performance of our

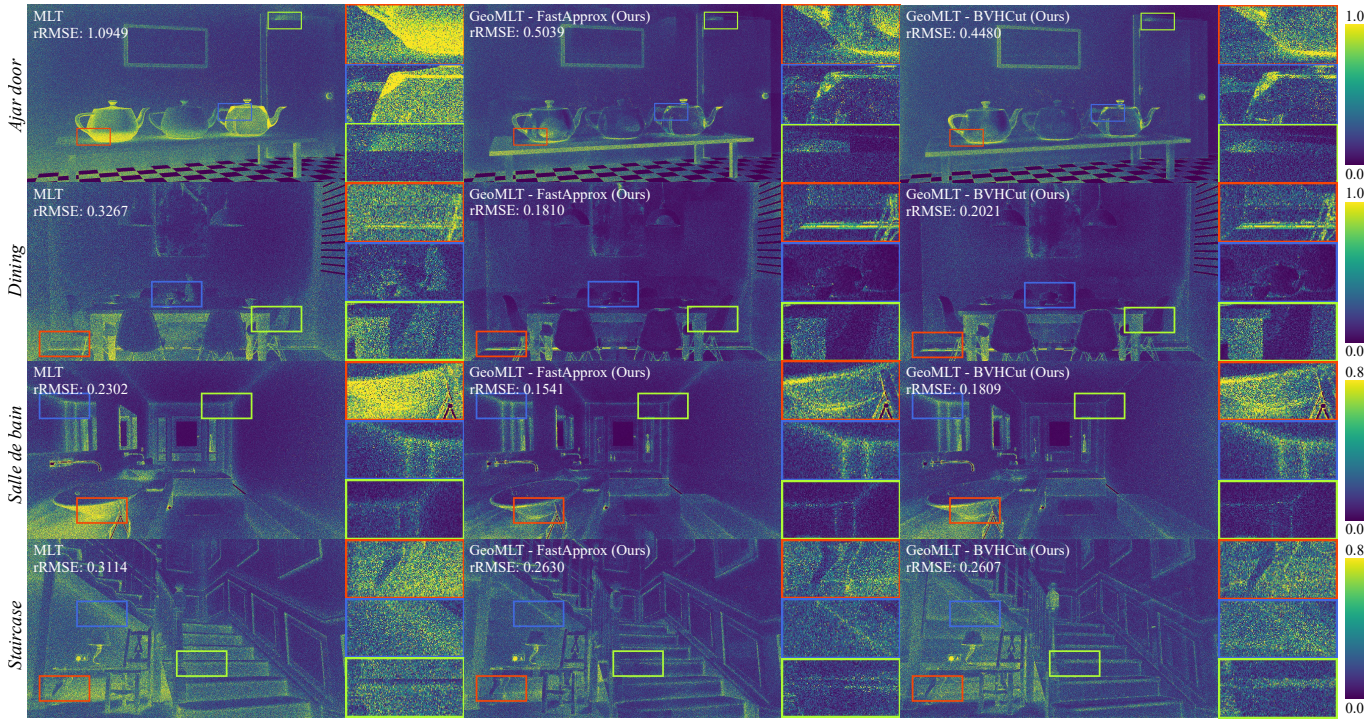

Fig. 12. Error distribution of the scenes (*Ajar door*, *Dining room*, *Salle de bain*, *Staircase*).

approach is similar to MLT. The effectiveness of our approach can also be observed in the pixel-wise error distribution (Fig. 12). We use the relative root mean square error (rRMSE) as an error metric for the images.

Fig. 11 (top) resembles the introductory example of Fig. 4 in a more practical scenario: the light enters through window blinds which are almost closed. As can be seen in Fig. 12, GeoMLT improves the error significantly compared to MLT. Furthermore, it becomes apparent that the effectiveness of the algorithm is also due to difficult visibility under the table and close to the dishes on the table (for instance the error on the back wall does not change much).

Fig. 11 (middle) shows an equal-time comparison in the *Salle de bain* scene, which contains several different materials including diffuse, specular, and glossy. The scene is illuminated with a large area light source directly visible from the camera. Unlike the *Ajar door* scene, the major part of the scene is directly illuminated by the light source. Even in such a simple lighting situation, our proposed approach can improve the image quality – especially in highly occluded parts or near object boundaries. The error distribution as compared to the reference is shown in Fig. 12. We can observe that the error distribution of the parts that can be seen through the mirror object (blue inset) has a characteristic that is different from the part that can be seen directly from the sensor (green inset). This is because the maximum cone angle estimation is applied only to the segments starting from a non-specular surface. In other words, no estimation happens for the vertices associated with the specular surface. In this sense, this part of the scene shows a failure case of our approach.

Fig. 11 (bottom) shows equal-time renders of the *Staircase* scene. This scene has a similar characteristic as the *Salle de bain*. Many parts of the scene are also directly illuminated and some parts, e.g., under the stairs, are indirectly lit by light bouncing through moderately complex occlusion. The error distribution is shown in Fig. 12. Again, our approach improves the parts of the scene near geometric discontinuities (such as in the insets). However, in this scene we can also observe a part where naive MLT is better than GeoMLT: on the pillar on the left. This is because tracing a cone from the camera to the pillar results in a big opening angle, since there is no nearby geometric obstruction. In fact, the opposite is the case: a large mutation step will jump off the pillar instead of exploring it, because there is no nearby geometry. Thus, our cone estimation is suboptimal in this part of the scene, yet the overall rRMSE is still better than for standard MLT.

Accuracy of the BVHCut Approximation. Fig. 13 shows cone angle estimates for primary rays. The values are computed from the segment obtained by the intersection between the primary ray and the surface. The colors correspond to the values in $[\tau_{\min}, \tau_{\max}]$. The results with the exact cone angle calculation (Eq. 7) are shown in the first row. The *BVHCut* (Sec. 5.2) version looks similar to the exact case, because subdividing the AABBs results in a quite accurate estimation. On the other hand, *FastApprox* (Sec. 5.3) shows some structured artifacts. We want to note that even this simple approach can coarsely capture the characteristics of the exact solution.

Dependency on the Acceleration Structure. Because the approximation depends on the acceleration structure used for ray tracing, we

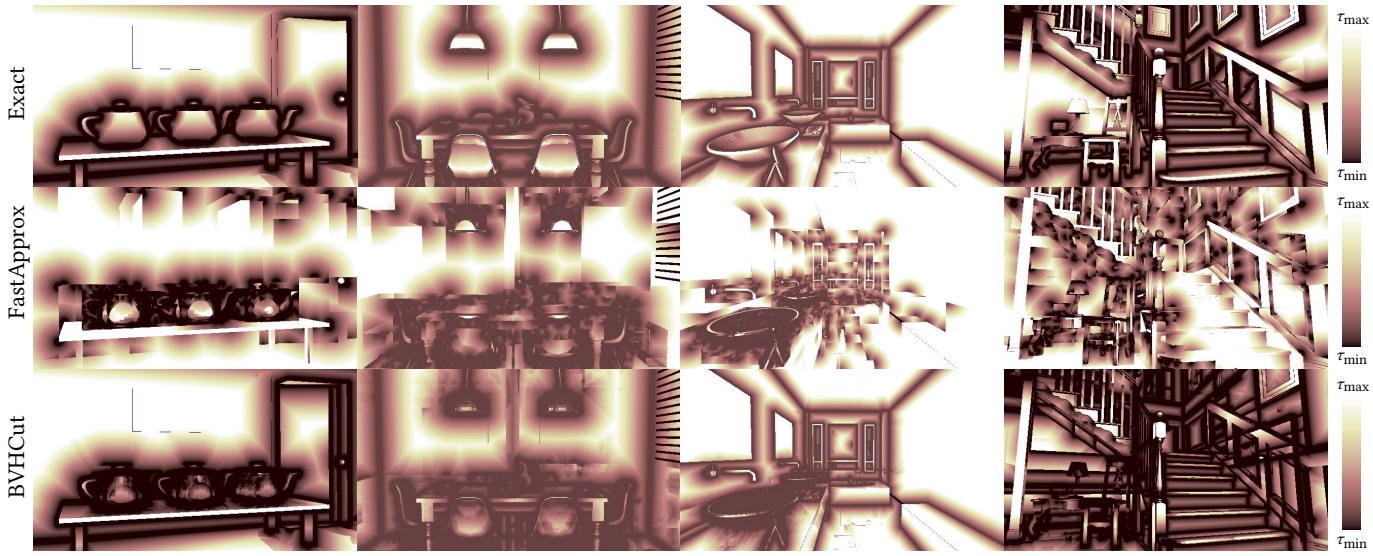

Fig. 13. Visualization of the cone angle distribution over the image plane. The color ramp ranges from $\tau_{min} = 10^{-2}$ to $\tau_{max} = 5 \cdot 10^{-1}$.

evaluate the image error with respect to this data structure. Fig. 14 shows how the error distribution changes if the scene is rotated around the up vector. The rotation of the scene forces the acceleration structure to change because the axis aligned bounding boxes change. In both cases, we can observe visible changes in the error. For instance, in *FastApprox*, we can find a discontinuity on the back wall. Although still noticeable, *BVHCut* can alleviate these negative effects because of the similarity to the exact cone angle computation (which depends on the geometry, but not the acceleration structure).

Quality of Exploration. After the burn-in phase which removes start-up bias, the samples from a Markov chain follow the central limit theorem. The quality of the exploration of a Markov chain can be quantified by looking at the autocorrelation of the samples in the chain. This autocorrelation affects the effective sample size (ESS), which then reduces the Markov chain Monte Carlo standard error MCMC-SE = $\sqrt{\text{Var}/\text{ESS}}$ much like the number of samples in plain Monte Carlo would [Betancourt 2017, Sec. 2.2]. The autocorrelation is an intuitive measure for how clumpy the appearance of the sampling scheme will be. Since it is not straight forward to estimate [Geyer 1992], we give RMSE values for equal sample comparisons instead. These numbers shown in Fig. 15 relate directly to the MCMC-SE.

Differently-Sized Geometry. Fig. 16 shows equal-time renders of the *Tree* scenes with contain geometries of different details (larger trunk, smaller branches). The scene is illuminated by an area light, with (bottom) and without (top) a diffuse reflector behind the tree model. In both cases our approach outperforms MLT, however, the performance gain is greater for the scene without the reflector. In the scene with the reflector, MLT is likely to make a valid mutation for the path segment connecting a point on the sensor and a point on the reflector. This is because a perturbed path segment can jump over the branches with thin geometries. Although we can observe that even in the scene with the reflector the rendering of tree branches is still better with our approach, the rendering of the background compensates the error as a whole as the reflector covers half of the image. It can be interesting for future work to determine how to best balance mutations which are aware and unaware of the geometry.

8 DISCUSSION

Parameters. Our method has several parameters to control its behavior. Similar to most rendering algorithms, the selection of these parameters affects the final result. The performance of our approach also obviously depends by the configuration of the scene geometry, because the maximum cone angle estimation depends on the acceleration structure. Although we leave a detailed error analysis for future work, we observed that looser bounding volumes can introduce larger errors in the estimation. This observation implies that bounding volumes have to be tight for our method to work best, however, this is generally desirable for acceleration structures.

Cone Angle Estimation. We have tested variants of cone angle estimations, where *BVHCut* and *FastApprox* can be seen as corner cases (elaborate vs. very coarse) – which, however, provide quite similar performance. Still there might be more efficient and effective solutions, possibly depending on the exact accleration structure and its implementation.

Limitations. As pointed out in the discussion about the *Salle de bain* scene (Fig. 11, middle), our approach currently does not handle the estimation of cone angles for segments passing through specular surfaces. Instead, our approach focuses on the perturbation of a single path segment starting from a diffuse surface. This becomes visible when a part of the scene is observed through a mirror as in the *Salle de bain* scene.

Another limitation is potentially visible discontinuities in the error distribution of the image, e.g., on the door in the *Ajar door* scene

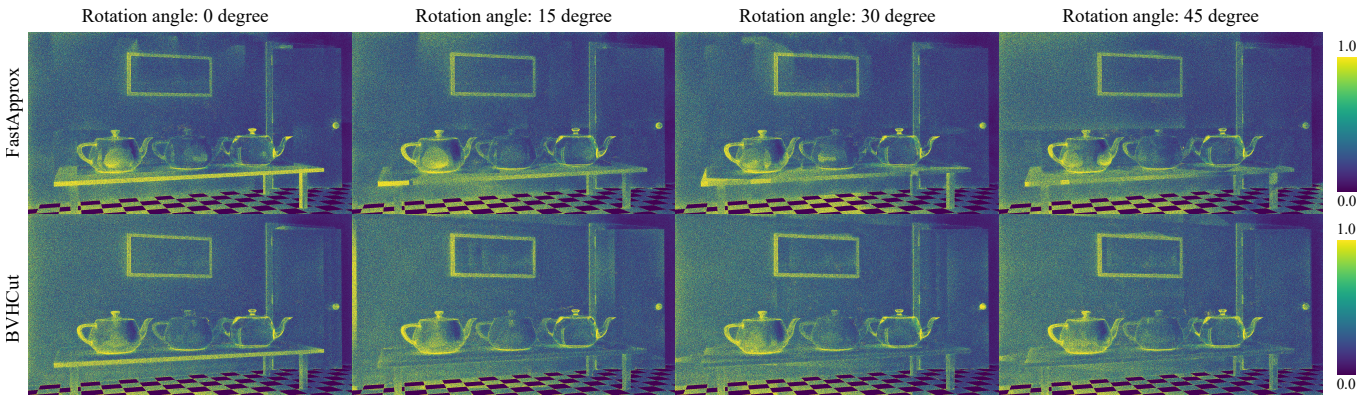

Fig. 14. Error plots (rRMSE per pixel) with rotated scene geometry, resulting in rotated AABBs and thus a different acceleration structure. The BVHCut version is more stable under rotation of the input, because of the adaptive subdivision.

(Fig. 12, top). Although *BVHCut* alleviates this issue to some extent, these discontinuities stem from the AABBs of the acceleration structure and the clipping of the estimated cone angle.

We also note that there are situations where our approach might not outperform MLT due to the overhead of cone angle estimation. If the estimated cone angle is clamped to $\tau_{\max}$, our approach essentially falls back to the original perturbation. In this case, the additional computation has been superfluous.

Reusing Acceleration Structure. Our approach reuses the acceleration structure for ray-scene intersection. Although using a separate acceleration structure to represent only the empty spaces would be an interesting future work, we decided not to use such a structure since the use of the acceleration structure enables us to perform ray-scene intersection and cone fitting queries at the same time. It is important to note that we need two cone fitting queries for each perturbation of a segment, one for the forward transition probability and one for the reverse. We also need a ray-scene intersection query with the perturbation. Alg. 2 shows how it is achieved efficiently, which would not be possible with a separate data structure.

Future work. Our cone estimation scheme would be more efficient if cones could anisotropically adapt to the shape of the geometry. For instance, in the *Ajar door* scene, since the rooms are connected with a narrow gap that extends all the way to the ceiling, having vertically elongated cones would better with the visibility term in this case. This could be achieved by estimating cone angles for multiple directions orthogonal to a path segment.

Our approach focuses only on surface rendering at the moment. Supporting participating media would be an interesting extension, although it involves additional difficulties concerning distance sampling in a volume for example.

Although a combination of mutation techniques in different state spaces is possible [Bitterli et al. 2017; Otsu et al. 2017; Pantaleoni 2017], designing geometry-aware mutations in primary sample space is a fascinating candidate for future work. For instance, it would be interesting if we could do geometry-aware mutations directly in the primary sample space, which would enable us to combine with other local adaptation approaches, such as the approach by Li et al. [2015].

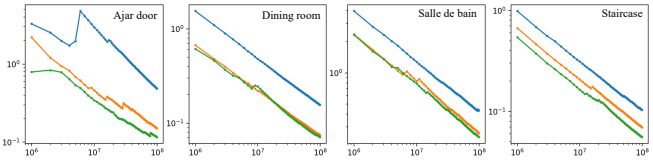

Fig. 15. Asymptotic error behavior: RMSE over the number of mutations (blue: MLT, orange: FastApprox, green: BVHCut).

9 CONCLUSION

We presented a mutation strategy which adaptively changes the mutation step size according to the geometry of the scene. Our method restricts perturbations of path segments such that nearby geometry is not intersected, as this would always result in rejected proposals. We introduced fast, approximate algorithms to estimate the maximum perturbation angle, which reuse the very same acceleration structure which is already present for ray casting. We demonstrated that our approach can greatly improve the exploration performance of a Markov chain. Our perturbation strategy has been designed with small geometric features in mind, such as door slits or keyholes where the light shines through. However, as our results show, it also reduces noise near geometric edges, such as the one between the floor and the back wall in the Ajar door scene. We believe that using information about geometric visibility has great potential and can be used to ameliorate many other cases of inefficient mutations due to geometric constraints.

ACKNOWLEDGMENTS

We would like to thank the reviewers for their insightful comments. The *door* scene is originally made by Miika Aittala, Samuli Laine, and Jaakko Lehtinen. We thank the blendswap.com artist Wig42 for the *Dining room* and *Staircase* scene and nacimus for the *Salle de bain* scene. These scenes are converted and distributed by Benedikt Bitterli. The *Tree* scene uses the Horse Chestnut Tree model courtesy of Xfrog Inc. This project was partly funded by JSPS KAKENHI Grant Number 15H05308.

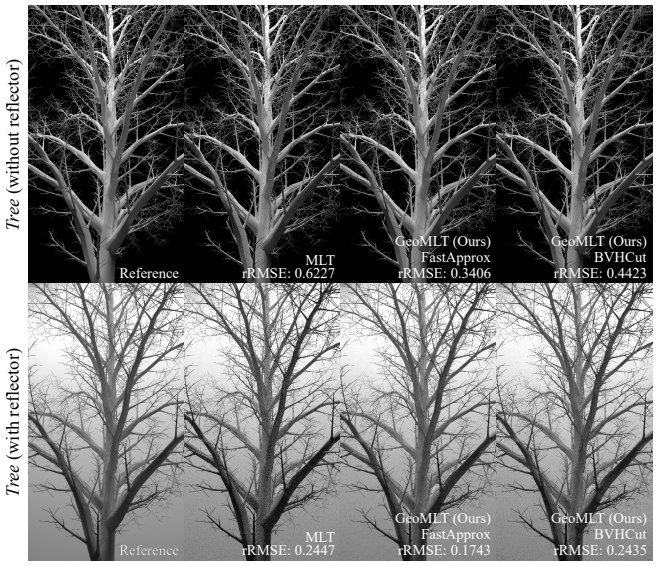

Fig. 16. Equal-time comparisons of the scenes with different sized geometries. The scenes contain same tree model and illuminated by area lights with (bottom) and without (top) diffuse reflector behind the tree model.

REFERENCES

John Amanatides. 1984. Ray Tracing with Cones. *Computer Graphics (Proc. SIGGRAPH)* 18, 3 (1984), 129–135.

James Arvo. 1986. Backward ray tracing. In *Developments in Ray Tracing, ACM SIGGRAPH Course Notes*. 259–263.

Michel Betancourt. 2017. A Conceptual Introduction to Hamiltonian Monte Carlo. *ArXiv e-prints* (Jan. 2017). arXiv:stat.ME/1701.02434

Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. 2017. Reversible Jump Metropolis Light Transport Using Inverse Mappings. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37, 1, Article 1 (2017), 12 pages.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive Indirect Illumination Using Voxel Cone Tracing. *Computer Graphics Forum (Proc. Pacific Graphics)* 30, 7 (2011).

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

Charles J. Geyer. 1992. Practical Markov Chain Monte Carlo. *Statist. Sci.* 7, 4 (11 1992), 473–483. https://doi.org/10.1214/ss/1177011137

Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed Metropolis Light Transport. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4, Article 100 (2014).

Johannes Hanika, Anton S. Kaplanyan, and Carsten Dachsbacher. 2015. Improved Half Vector Space Light Transport. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 34, 4 (2015), 65–74.

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

Wenzel Jakob and Stephen Marschner. 2012. Manifold exploration: a Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31, 4, Article 58 (2012).

James T. Kajiya. 1986. The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH '86)* 20, 4 (1986), 143–150.

Anton Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 2014. The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33, 4 (2014), 1–13.

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

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Compugraphics '93*. 145–153.

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 Transactions on Graphics (Proc. SIGGRAPH Asia)* 34, 6 (2015), 209:1–209:13.

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

Benjamin Mora. 2011. Naive Ray-tracing: A Divide-and-conquer Approach. *ACM Transactions on Graphics* 30, 5, Article 117 (2011), 12 pages.

Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. 2017. Fusing State Spaces for Markov Chain Monte Carlo Rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36, 4, Article 74 (2017), 10 pages.

Jacopo Pantaleoni. 2017. Charted Metropolis Light Transport. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 36, 4, Article 75 (2017), 14 pages.

David Roger, Ulf Assarsson, and Nicolas Holzschuch. 2007. Whitted Ray-tracing for Dynamic Scenes Using a Ray-space Hierarchy on the GPU. In *Proc. Eurographics Symposium on Rendering*. 99–110.

Eric Veach and Leonidas J. Guibas. 1994. Bidirectional Estimators for Light Transport. In *Proc. Eurographics Workshop on Rendering*. 147–162.

Eric Veach and Leonidas J. Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. *Proc. SIGGRAPH '95* (1995), 419–428.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *SIGGRAPH '97*. 65–76.