Manifold Exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport

Expanded Technical Report

Wenzel Jakob Steve Marschner

Cornell University

May 7, 2012

Abstract: It is a long-standing problem in unbiased Monte Carlo methods for rendering that certain difficult types of light transport paths, particularly those involving viewing and illumination along paths containing specular or glossy surfaces, cause unusably slow convergence. In this paper we introduce Manifold Exploration, a new way of handling specular paths in rendering. It is based on the idea that sets of paths contributing to the image naturally form manifolds in path space, which can be explored locally by a simple equation-solving iteration. This paper shows how to formulate and solve the required equations using only geometric information that is already generally available in ray tracing systems, and how to use this method in in two different Markov Chain Monte Carlo frameworks to accurately compute illumination from general families of paths. The resulting rendering algorithms handle specular, near-specular, glossy, and diffuse surface interactions as well as isotropic or highly anisotropic volume scattering interactions, all using the same fundamental algorithm. An implementation is demonstrated on a range of challenging scenes and evaluated against previous methods.

Remark: This technical report follows the structure of the main paper. The main difference is that it contains additional discussion in Sections 3.3 and 6, a second appendix with C++ code, as well as MEMLT results in Section 8.

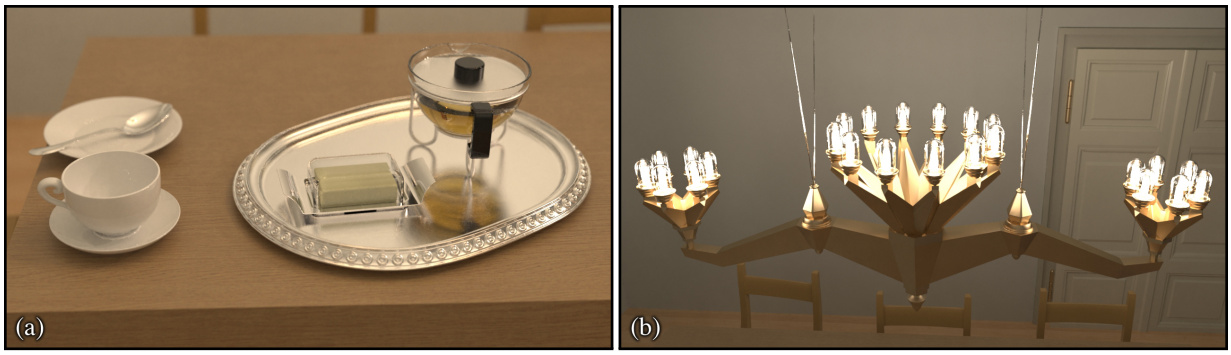


Figure 1: Two views of an interior scene with complex specular and near-specular transport, rendered using manifold exploration path tracing: **(a)** Refractive, reflective, and glossy tableware, **(b)** A rough brass luminaire with 24 glass-enclosed bulbs used to light the previous closeup.

1 Introduction

Certain classes of light paths have traditionally been a source of difficulty in conducting Monte Carlo simulations of light transport. A well-known example is specular-diffuse-specular paths, such a tabletop seen through a drinking glass sitting on it, a bottle containing shampoo or other translucent liquid, or a shop window viewed and illuminated from outside. Even in scenes where these paths do not cause dramatic lighting effects, their presence can lead to unusably slow convergence in renderers that attempt to account for all transport paths.

Furthermore, wherever ideally specular paths are troublesome, nearly specular paths involving glossy materials are also troublesome. They can be more problematic, in fact, because they elude special mechanisms designed to handle specular interactions. These glossy paths have become more important as material models have evolved, and we would prefer to handle them using natural generalizations of strategies for specular surfaces, rather than generalizations of strategies for diffuse surfaces.

Finding these paths efficiently is a key problem of light transport simulations. In this paper, we show how a Markov Chain Monte Carlo method can be used to efficiently render paths with illumination and/or viewing through arbitrary chains of specular or glossy reflections or refractions. The idea is that sets of paths contributing to the image naturally form manifolds in path space, and using a simple equation-solving iteration it is easy to move around on these manifolds. We show how to formulate and solve the equations required to find specular paths, using only geometric information that is already generally available in ray tracing systems. This solution method is then used to define a Markov Chain, which has the right properties to be applied in a Metropolis-Hastings algorithm to compute lighting through very general families of paths that can involve specular, near-specular, glossy, and diffuse surface interactions as well as isotropic or highly anisotropic volume scattering interactions.

We begin by discussing prior work on which our method is built, then in Section 3 we develop the theory of the *specular manifold*, used to handle interactions with ideal specular (polished) surfaces, and *offset specular manifolds*, which provide a graceful generalization to near-specular materials. In Section 5, we derive an algorithm to move from one path to another on a specular or offset specular manifold, use it to build an algorithm that generates Markov sequences in path

space, and show how this can be used in the Metropolis Light Transport or Energy Redistribution Path Tracing frameworks to provide methods for rendering scenes with any kind of light transport. We go on in the following sections to extend the theory and algorithm to the case of participating media, and we finish by showing results and comparisons in Section 8.

2 Prior work

Simulating light transport has been a major effort in computer graphics for over 25 years, beginning with the complementary approaches of finite-element simulation, or radiosity [Goral et al. 1984], and ray-tracing [Whitted 1980]. The introduction of Monte Carlo methods for ray tracing [Cook et al. 1984], followed by Kajiya's formulation of global illumination in terms of the Rendering Equation [Kajiya 1986], established the field of Monte Carlo global illumination. Unbiased sampling methods, in which each pixel in the image is a random variable with an expected value exactly equal to the solution of the Rendering Equation, started with Kajiya's original path tracing method and continued with bidirectional path tracing [Lafortune and Willems 1993; Veach and Guibas 1994], in which light transport paths can be constructed partly from the light and partly from the eye, and the seminal Metropolis Light Transport [Veach and Guibas 1997] algorithm, which uses bidirectional path tracing methods in a Markov Chain Monte Carlo framework.

Various two-pass methods use a particle-tracing pass that sends energy out from light sources in the form of "photons" that are traced through the scene [Shirley et al. 1995; Jensen 1996; Walter et al. 1997]. and stored in a spatial data structure. The second pass then renders the image using ray tracing, making use of the stored particles to estimate illumination by density estimation. [Jensen and Christensen 1998; Jarosz et al. 2008; Hachisuka and Jensen 2009].

Photon mapping and other two-pass methods are characterized by storing an approximate representation of some part of the illumination in the scene, which requires assumptions about the smoothness of illumination distributions. On one hand, this enables rendering of some modes of transport that are difficult for unbiased methods, since the exact paths by which light travels do not need to be found; separate paths from the eye and light that end at nearby points suffice under assumptions of smoothness. However, this smoothness assumption inherently leads to smoothing errors in images: the results are biased, in the Monte Carlo sense. Glossy-glossy transport, without a sufficiently diffuse surface on which to store photons, is challenging to handle with photon maps, since large numbers of photons must be collected to adequately sample position-direction space. Some photon mapping variants avoid this by treating glossy materials as specular, but this means that the the resulting method increasingly resembles path tracing as the number of rough surfaces in the input scene grows.

2.1 Specular reflection geometry

Separate from work on global illumination algorithms, various research has examined the properties of specular reflection paths. Mitchell and Hanrahan [1992] devised a method to compute irradiance from implicitly defined reflectors, using Fermat's principle with interval Newton's method to locate all reflection paths from a source to a point. Walter et al. [2009] proposed a related method that computes the singly scattered radiance within a refractive object with triangle mesh boundaries. Like these works, our method searches for specular paths. But because it does so within the *neighborhood* of a given path, it avoids the complexities and constraints entailed by a full global

search. Another difference is that our manifold formalism can be used to build a fully general rendering system that is not limited to the specular paths that prompted its design.

The widely used method of ray differentials for texture antialiasing [Igehy 1999] also reasons about the local structure of a set of reflected paths—in this case, paths from the eye. Igehy's approach requires elementary local differential information only, in the form of derivatives of surface normals, and does not require global surface descriptions as Mitchell and Hanrahan's method does. Manifold exploration requires the same geometric information, and can thus be implemented in most modern ray tracing sytems.

The analysis of reflection geometry presented by Chen and Arvo [2000a; 2000b] is closest to the mathematics underlying our proposed methods. Their work relies on a characterization of specular paths via Fermat's principle, which asserts that light travels along paths whose optical length (i.e. the propagation time) constitutes a local extremum amongst neighboring paths. Using Lagrange multipliers, the authors derive a *path Jacobian* and *path Hessian* with respect to perturbations of the endpoint of a path and use it to accelerate the interactive display of reflections on curved surfaces.

Our local characterization of specular paths is equivalent to Fermat's principle, and the path Jacobian is related to the derivatives that we propose to use to define tangent spaces to the specular manifold while solving for path transitions. However, the use of this derivative and the goals of the research are entirely different: in their case, estimating changes to viewing paths, and in our case, tracking the evolution of specular paths in a very general context, as part of an unbiased rendering system.

A less related but relevant idea is integrating over continuous paths in volume rendering applications, known as the path integral formulation of radiative transfer [Premoze et al. 2003]. This work, with its implications for the concentration of transport in path space, suggests the possibility of using MCMC to integrate over multiple-scattering paths in volumes, as discussed in Section 7.

2.2 Markov Chain Monte Carlo in rendering

The Metropolis Light Transport algorithm mentioned above introduced the tools of Markov Chain Monte Carlo (MCMC) to rendering. A Markov chain is a sequence of points in a state space in which the probability of a state appearing at a given position in the sequence depends only on the previous state. The basic idea of MCMC, first proposed by Metropolis et al. [1953], is to define a Markov chain that has the function to be integrated as its stationary distribution, meaning that if the chain is run for a long time the distribution of states it visits will be proportional to the desired distribution. The following brief introduction follows [Liu 2001].

Defining a Markov chain amounts to defining a *transition rule*: a process for selecting a new state $\mathbf{x}_+$ randomly, in a way that depends on the current state $\mathbf{x}$. Metropolis et al. provided a way to take a transition rule that may not produce the desired stationary distribution $\pi(\mathbf{x})$ and turn it into one that does. Given a method for sampling a *proposal distribution* $T(\mathbf{x}, \mathbf{x}')$, the Metropolis transition rule operates in two steps:

1. Choose $\mathbf{x}'$ according to the probability distribution $T(\mathbf{x}, \mathbf{x}')$.
2. $\mathbf{x}_+ = \begin{cases} \mathbf{x}' & \text{with probability } \min(1, \pi(\mathbf{x}')/\pi(\mathbf{x})) \\ \mathbf{x} & \text{otherwise} \end{cases}$

In step 1 we say $\mathbf{x}'$ is *proposed* as the next state, and in step 2 it is either *accepted* and becomes the next state, or it is *rejected* and the next state repeats the previous one. The probability $\min(1, \pi(\mathbf{x}')/\pi(\mathbf{x}))$ is known as the *acceptance probability*.

If this chain is able to pass from any state in the domain to another using a finite expected number of steps, and if the length of this journey is in a sense "irregular" (i.e. aperiodic), it is referred to as *ergodic*, and the chain's limiting distribution is guaranteed to converge to π. This relatively mild criterion is usually satisified by choosing a transition rule with global support.

The original Metropolis algorithm only works when $T(\mathbf{x}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x})$. Hastings [1970] proposed a new acceptance probability:

$$r(\mathbf{x}, \mathbf{x}') = \min\left\{1, \frac{\pi(\mathbf{x}')T(\mathbf{x}', \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x}, \mathbf{x}')}\right\} \tag{1}$$

which relaxes the symmetry restriction to one of symmetric support: $T(\mathbf{x}, \mathbf{x}')$ must be nonzero exactly when $T(\mathbf{x}', \mathbf{x})$ is nonzero. The Metropolis–Hastings algorithm is the starting point for MCMC rendering methods.

In the rendering context, the state space is the space of all paths through the scene, points in the space are paths, and the desired probability distribution over paths is proportional to their contribution to the rendered image (i.e. the amount of illumination they carry to the camera). The final image is the projection of the path distribution into the image plane.

At the core of an MCMC rendering algorithm is an implementation of a transition rule, and any rule with symmetric support is admissible. But to avoid very low acceptance probabilities, which lead to poor performance, it is desirable for the transition probability to approximate the contribution: that is, paths with more light flowing along them should be chosen more often. Veach's [1997] transition rule is based on a set of *mutations* that change the structure of the path and *perturbations* that move the vertices by small distances while preserving the structure, both using the building blocks of bidirectional path tracing to sample paths. Using his transition rule to run long chains leads to the Metropolis Light Tranport algorithm (MLT). Kelemen et al. [2002] later proposed a transition rule based on changing the random numbers used to sample paths by bidirectional path tracing.

Considerable research activity has extended Metropolis light transport in various ways. Pauly et al. [2000] proposed a perturbation rule for rendering participating media with single scattering. Other projects include Metropolis Instant Radiosity [Segovia et al. 2007], Population Monte Carlo rendering [Lai et al. 2007], and Replica Exchange light transport [Kitaoka et al. 2009]. Recently, two groups [Chen et al. 2011; Hachisuka and Jensen 2011] have combined the transition rule of Kelemen et al. with photon mapping to obtain robust methods based on density estimation.

However, to generate proposals, all of these algorithms ultimately rely on local path sampling strategies (i.e. path tracing). Specifically, they choose the next interaction vertex along a light path by sampling from a directional distribution associated with the current vertex, followed by an intersection search. Our method introduces a new kind of transition rule with different properties.

The original MLT algorithm and subsequent variants all render an image by running a Markov chain for a long (e.g. $> 10^6$) sequence of steps, and they guarantee ergodicity using a transition rule that can generate any path in the domain with some probability. The Energy Redistribution path tracing (ERPT) technique [Cline et al. 2005], which is readily adapted to work with our method, is an interesting departure from that approach. It draws on the property that the

Metropolis-Hastings algorithm (1) preserves the stationary distribution of samples even if the underlying transition rule is *not* ergodic (e.g. when it cannot reach parts of path space). To obtain coverage, ERPT first samples a large set of paths via path tracing and runs Markov chains for short bursts ($\approx 10^3$ steps) starting at each sample. The relaxation of the ergodicity requirement makes it possible to explore paths in a very local fashion.

3 Path space manifolds

Manifold exploration is a technique for integrating the contributions of sets of specular or near-specular illumination paths to the rendered image of a scene. The general approach applies to surfaces and volumes and to ideal and non-ideal (glossy) specular surfaces. In this section we begin by examining the manifold defined by ideal specular reflection or refraction, in the setting of surfaces without participating media. In the following section we develop this theory into a rendering method for scenes combining ideal specular surfaces with fairly diffuse surfaces. We will then go on to generalize the method to glossy surfaces, then to generalize the theory to encompass participating media and to extend the method to handle media with both isotropic and highly directional scattering.

3.1 Path space

The resulting techniques are all based on the path integral formulation of light transport, described by Veach [1997] and others, in which the value of each pixel in the image is an integral of a *contribution function* over *path space*. Denoting the union of all surfaces in the scene by $\mathcal{M}$, each sequence $\mathbf{x}_1 \dots \mathbf{x}_n$ of at least two points in $\mathcal{M}$ is a path along which light may travel. Thus path space is

$$\mathcal{P} = \bigcup_{n=2}^{\infty} \mathcal{P}_k \tag{2}$$

$$\mathcal{P}_n = \{\mathbf{x}_1 \dots \mathbf{x}_n \mid \mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{M}\}\,. \tag{3}$$

As detailed by Veach, the value of pixel j is

$$I_j = \int_{\mathcal{P}} f_j(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}})$$

where $\bar{\mathbf{x}} = \mathbf{x}_1 \dots \mathbf{x}_n$ denotes an element of $\mathcal{P}$ and μ is the product measure derived from the area measure on $\mathcal{M}$. The contribution function f_j is a product of terms, one for each vertex and each edge of the path:

$$f_j(\mathbf{x}_1 \cdots \mathbf{x}_n) = L_e(\mathbf{x}_1 \to \mathbf{x}_2) \left[\prod_{k=2}^{n-1} G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)\, f_s(\mathbf{x}_{k-1} \to \mathbf{x}_k \to \mathbf{x}_{k+1})\right] G(\mathbf{x}_{n-1} \leftrightarrow \mathbf{x}_n)\, W_e^{(j)}(\mathbf{x}_{n-1} \to \mathbf{x}_n). \tag{4}$$

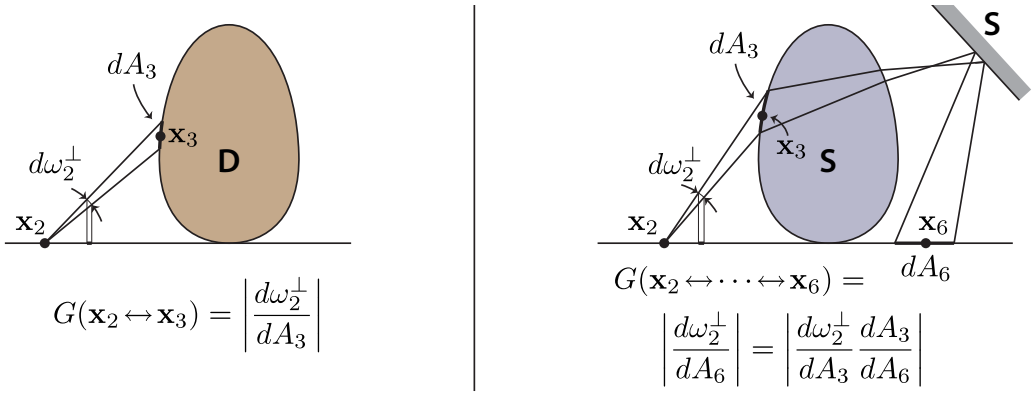


Figure 2: The geometry factor (left) and the generalized geometry factor (right) are both derivatives of projected solid angle at one end with respect to area at the far end.

Here L_e is emitted radiance, $W_e^{(j)}$ is importance for the jth pixel, f_s is the BSDF at $\mathbf{x}_k$ for the geometry defined by $\mathbf{x}_{k-1}$, $\mathbf{x}_k$, and $\mathbf{x}_{k+1}$, and G is the *geometry factor*:

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = \frac{|N(\mathbf{x}) \cdot \overrightarrow{\mathbf{xy}}| \, |N(\mathbf{y}) \cdot \overrightarrow{\mathbf{yx}}|}{\|\mathbf{x} - \mathbf{y}\|^2} V(\mathbf{x} \leftrightarrow \mathbf{y}) \tag{5}$$

where $N(\mathbf{a})$ is the surface normal at $\mathbf{a}$ and $V(\mathbf{x} \leftrightarrow \mathbf{y})$ is the visibility function.

3.2 Motivating examples

This formulation is the basis for several rendering methods including bidirectional path tracing and Metropolis light transport. However, in the presence of ideal specular reflection, some difficulties arise, which are normally sidestepped in the transition from theory to algorithm, but which we prefer to confront directly. When some surface interactions are specular, the entire contribution to the path space integral is from paths that obey specular reflection or refraction geometry, and the set of such paths is lower in dimension than the full path space. For instance, consider a family of paths of the form LDSDE (in Heckbert's [1990] notation) with one specular reflection vertex. These paths belong to the $\mathcal{P}_5$ component of $\mathcal{P}$, but the paths that contribute all have the property

$$\left(\overrightarrow{\mathbf{x}_3\mathbf{x}_2} + \overrightarrow{\mathbf{x}_3\mathbf{x}_4}\right) \parallel N(\mathbf{x}_3),$$

that is, the half-vector at $\mathbf{x}_3$ is in the direction of the normal. This places two constraints on the path, meaning that all contributing paths lie on a manifold of dimension 8 embedded in $\mathcal{P}_5$, which is of dimension 10. The integral is more naturally expressed as an integral over the manifold, rather than as an integral over the whole path space.

To compute illumination due to specular paths, we use a local parameterization of the manifold in terms of the positions of all nonspecular vertices on the path:

$$\iiiint_{\mathcal{M}^4} f(\mathbf{x}_1 \ldots \mathbf{x}_5) d\mathbf{x}_1 d\mathbf{x}_2 d\mathbf{x}_4 d\mathbf{x}_5$$

Note the missing integral over $\mathbf{x}_3$, the specular vertex. The contribution function f still has

the same form, a product of terms corresponding to vertices and edges of the path, but the BSDF values at the specular vertex are replaced by (unitless) specular reflectance values, and the geometry factors for the two edges involving the specular vertex are replaced by a single generalized geometry factor that we will denote $G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3 \leftrightarrow \mathbf{x}_4)$.

The standard geometry factor for a non-specular edge is the derivative of projected solid angle at one vertex with respect to area at the other vertex, and the generalized geometry factor is defined analogously: the derivative of solid angle at one end of the specular chain with respect to area at the other end of the chain, considering the path as a function of the positions of the endpoints. Figure 2 illustrates this for a more complex path involving a chain of three specular vertices. We will explain below how G can be easily computed from the differential geometry of the specular manifold.

3.3 Specular manifold geometry

In the general case, each path of length k belongs to a class in $\{D, S\}^k$ based on the classification of each of its vertices. (In this scheme point or orthographic cameras, and point or parallel lights, are denoted S, while finite-aperture cameras and area lights are D.) Each S surface vertex has an associated constraint that involves its position and the position of the preceding and following vertices:

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{0}$$

The constraint function computes a half-vector at vertex i and projects it into the shading tangent space; the resulting 2-vector is zero when the half-vector is parallel to the normal. By making use of the generalized half-vector of Walter et al. [2007], both reflection and refraction can be handled by a single constraint function:

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = T(\mathbf{x}_i)^T h(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}) \tag{6}$$

$$h(\mathbf{x}, \mathbf{v}, \mathbf{w}) = \frac{\eta(\mathbf{x}, \mathbf{v})\mathbf{v} + \eta(\mathbf{x}, \mathbf{w})\mathbf{w}}{\|\eta(\mathbf{x}, \mathbf{v})\mathbf{v} + \eta(\mathbf{x}, \mathbf{w})\mathbf{w}\|} \tag{7}$$

where $T(\mathbf{x})$ is a matrix whose columns form a basis for the shading tangent plane at $\mathbf{x}$, and $\eta(\mathbf{x}, \mathbf{v})$ denotes the refractive index associated with the ray $(\mathbf{x}, \mathbf{v})$. This generalized geometry factor is related to the "extended form factor" discussed by Sillion and Puech [1989].

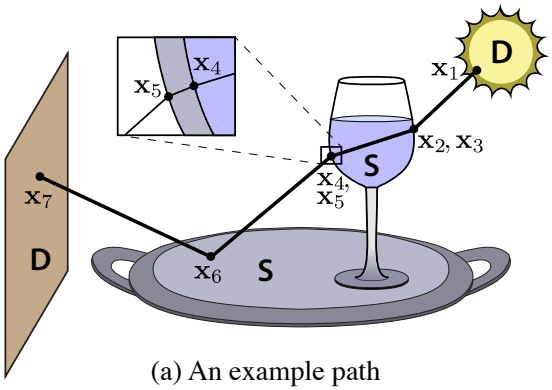
Specular endpoint vertices also introduce constraints that depend on their type: for instance, the position of a vertex on a point emitter must remain fixed (e.g. $\mathbf{x}_1 = \text{const}$). When the emitter is directional, it is the outgoing direction that is constrained, and so on.

From now on, we implicitly identify each vertex $\mathbf{x}_i$ with an associated point in $\mathbb{R}^2$ using local parameterizations of $\mathcal{M}$. These parameterizations may be defined on arbitrarily small neighborhoods, since only their derivatives are relevant to what follows.

With this, the constraints for a length-n path with p specular vertices can be stacked together into a function $C : \mathbb{R}^{2n} \to \mathbb{R}^{2p}$, and the specular manifold is simply the set

$$\mathcal{S} = \{\bar{\mathbf{x}} \mid C(\bar{\mathbf{x}}) = \mathbf{0}\} \tag{8}$$

Expressing $\mathcal{S}$ using a constraint in this way makes it convenient to work with neighborhoods of a particular path. The Implicit Function Theorem [Spivak 1965] guarantees the existence of a

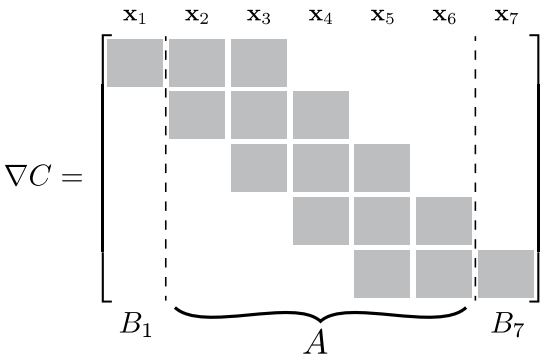


(a) An example path

$$C(\bar{\mathbf{x}}) = 0 \quad \text{where}$$

$$C = \begin{bmatrix} \mathbf{c}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \\ \vdots \\ \mathbf{c}_6(\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7) \end{bmatrix}$$

(b) Associated constraints



(c) Constraint Jacobian

$$\frac{\partial \begin{bmatrix} \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_6 \end{bmatrix}}{\partial \mathbf{x}_1} = -A^{-1}B_1$$

$$\frac{\partial \begin{bmatrix} \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_6 \end{bmatrix}}{\partial \mathbf{x}_7} = -A^{-1}B_7$$

(d) Tangent space

Figure 3: The linear system used to compute the tangent space to the specular manifold, also known as the derivative of a specular chain with respect to its endpoints.

parameterization of the manifold, in the neighborhood of any path $\bar{\mathbf{x}}$ that is nonsingular (in the sense explained below). This parameterization is a function $q : \mathbb{R}^{2(n-p)} \to \mathbb{R}^{2p}$ that determines the positions of all the specular vertices from the positions of all the nonspecular vertices. Furthermore, the derivative of q, which gives us the tangent space to the manifold at $\bar{\mathbf{x}}$, is simple to compute from the derivative of C.

For the specifics we restrict ourselves to the case of a single chain of specular vertices with non-specular vertices (surfaces, cameras, or light sources) at the ends, which suffices to cover most cases by applying it separately to mutiple specular chains along a path. Paths with specular endpoints are handled with simple variations of this scheme. Number the vertices in the chain $\mathbf{x}_1, \ldots, \mathbf{x}_k$, with $\mathbf{x}_1$ and $\mathbf{x}_k$ being the (non-specular) endpoints of the segment and the remaining $k-2$ vertices being specular. In this case $C : \mathbb{R}^{2k} \to \mathbb{R}^{2(k-2)}$, and the derivative ∇C is a matrix of $k-2$ by k 2-by-2 blocks, with a tridiagonal structure (Figure 3).

The Implicit Function Theorem gives us a parameterization of the manifold in terms of any 2 vertices, and if we pick $\mathbf{x}_1$ and $\mathbf{x}_k$ this simply says that the path, in a neighborhood of the

current path[1], is a function of the two endpoints. Furthermore, it also tells us the derivative of that parameterization, which is to say, the derivative of all the specular vertices' positions with respect to the positions of the endpoints. If we block the derivative ∇C, as shown in the figure, into 2-column matrices B_1 and B_k for the first and last vertices and a square matrix A for the specular chain, then this tangent space to the manifold is

$$T_{\mathcal{S}}(\bar{\mathbf{x}}) = -A^{-1} \begin{bmatrix} B_1 & B_k \end{bmatrix}$$

This matrix is $k-2$ by 2 blocks in size, and each block gives the derivative of one vertex (in terms of its own tangent frame) with respect to one endpoint.

The matrix A fails to be invertible when the specular vertices are not even locally unique for given endpoints, which means that one endpoint is on a caustic for light emitting from the other endpoint.

We use $T_{\mathcal{S}}(\bar{\mathbf{x}})$ for two things: to navigate on the manifold and to compute the generalized geometry factor. The right two or left two columns of $T_{\mathcal{S}}(\bar{\mathbf{x}})$ are useful for updating the specular chain with $\mathbf{x}_1$ or $\mathbf{x}_k$ held fixed, respectively, as discussed in the next section.

The top-right or bottom-left block of $T_{\mathcal{S}}(\bar{\mathbf{x}})$ can be used to compute the generalized geometry factor as follows. Assuming orthonormal parameterizations[2], the determinant of the top-right block gives the ratio of an infinitesimal area at $\mathbf{x}_k$ to its reflection/refraction, as observed from $\mathbf{x}_1$, measured on the surface at $\mathbf{x}_2$. To convert this to a ratio of area at $\mathbf{x}_k$ to solid angle at $\mathbf{x}_1$, we multiply this determinant by the ordinary geometry factor $G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$; this product is the generalized geometry factor $G(\mathbf{x}_1 \leftrightarrow \cdots \leftrightarrow \mathbf{x}_k)$. More succinctly,

$$\begin{aligned} G(\mathbf{x}_1 \leftrightarrow \cdots \leftrightarrow \mathbf{x}_k) &= \left|P_2 A^{-1} B_k\right| G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\ &= \left|P_{k-1} A^{-1} B_1\right| G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k), \end{aligned} \tag{9}$$

where P_i is a 2 by $2(k-2)$ matrix that projects onto the two dimensions associated with vertex i.

A useful property of this framework is its reliance on local information that is easily provided in ray tracing-based rendering systems. To compute the blocks of the A and B matrices, we must have access to the partial derivatives of position and shading normal with respect to any convenient parameterization of the surfaces, along with the refractive indices of all objects. These are exactly the same quantities also needed to trace ray differentials through refractive boundaries, which is part of many mature ray tracing-based rendering systems. A consequence of the simple form of the constraint (8) is that our technique works with any object that can provide such local information, including implicitly defined shapes or triangle meshes with shading normals.

These theoretical results about the structure of the specular manifold can be used in an algorithm to solve for specular paths, which we discuss in Section 4.

A simple example: We now demonstrate how to compute the tangent space and generalized geometric term associated with a simple example path shown in Figure 4. This is a path with three vertices; $\mathbf{x}_1$ and $\mathbf{x}_3$ are planar endpoints used to parameterize the manifold, and $\mathbf{x}_2$ lies on a

[1]Because it is possible to have several separated specular paths joining two points, the parameterization cannot be global.

[2]If the parameterizations are not orthonormal, two additional determinants are required to account for the change in area.

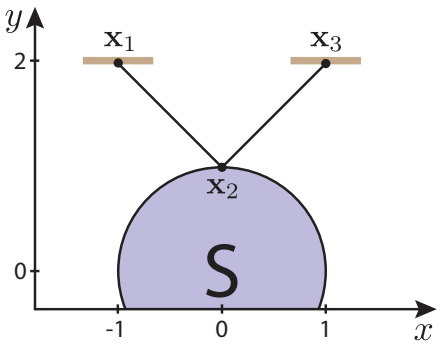


Figure 4: An example path with three vertices

specularly reflecting cylinder that stretches out along the z-axis.

Assuming that this path is encountered in a ray tracer similar to PBRT [Pharr and Humphreys 2004] or Mitsuba [Jakob 2010], the rendering system will associate an intersection data record with each vertex containing (amongst other things) the position and surface normal, as well as derivatives thereof along some parameterization. Suppose that the information provided is as follows:

$$\begin{array}{lllllll}
\mathbf{x}_1 = (-1,2,0), & \partial_u\mathbf{x}_1 = (-1,0,0), & \partial_v\mathbf{x}_1 = (0,0,1), & \mathbf{n}_1 = (0,-1,0), & \partial_u\mathbf{n}_1 = 0, & & \partial_v\mathbf{n}_1 = 0 \\
\mathbf{x}_2 = (0,1,0), & \partial_u\mathbf{x}_2 = (1,0,0), & \partial_v\mathbf{x}_2 = (0,0,1), & \mathbf{n}_2 = (0,1,0), & \partial_u\mathbf{n}_2 = (1,0,0), & & \partial_v\mathbf{n}_2 = 0 \\
\mathbf{x}_3 = (1,2,0), & \partial_u\mathbf{x}_3 = (1,0,0), & \partial_v\mathbf{x}_3 = (0,0,1), & \mathbf{n}_3 = (0,-1,0), & \partial_u\mathbf{n}_3 = 0, & & \partial_v\mathbf{n}_3 = 0
\end{array}$$

Note that the parameterizations above are locally orthonormal—this is generally preferable, since it simplifies many computations involving ∇C. In practice, we can simply apply a suitable linear transformation to the tangents and normal derivatives to to make them correspond to a locally orthonormal chart. Let us now compute the entries of ∇C.

Recall that the constraint associated with vertex $\mathbf{x}_2$ is

$$C(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_2) = T(\mathbf{x}_2)^T \left(\frac{\mathbf{x}_1-\mathbf{x}_2}{\|\mathbf{x}_1-\mathbf{x}_2\|} + \frac{\mathbf{x}_3-\mathbf{x}_2}{\|\mathbf{x}_3-\mathbf{x}_2\|}\right) \Bigg/ \left\| \frac{\mathbf{x}_1-\mathbf{x}_2}{\|\mathbf{x}_1-\mathbf{x}_2\|} + \frac{\mathbf{x}_3-\mathbf{x}_2}{\|\mathbf{x}_3-\mathbf{x}_2\|} \right\|$$

To compute derivatives of this expression, we can parameterize all needed ingredients to first order using the above data records, i.e.

$$\mathbf{x}_i(u_i,v_i) = \mathbf{x}_i + u_i(\partial_u\mathbf{x}_i) + v_i(\partial_v\mathbf{x}_i), \quad \mathbf{n}_i(u_i,v_i) = \mathbf{n}_i + u_i(\partial_u\mathbf{n}_i) + v_i(\partial_v\mathbf{n}_i)$$

$$T(u_2,v_2) = \begin{pmatrix} \partial_u\mathbf{x}_2 - \langle\partial_u\mathbf{x}_2, \mathbf{n}_2(u_2,v_2)\rangle\mathbf{n}_2(u_2,v_2) \\ \partial_v\mathbf{x}_2 - \langle\partial_u\mathbf{x}_2, \mathbf{n}_2(u_2,v_2)\rangle\mathbf{n}_2(u_2,v_2) \end{pmatrix}$$

The rest is just a big nested application of the product rule. When differentiating C, we get two terms: one which accounts for changes of the tangent frame at $\mathbf{x}_2$, with h held constant, and one which accounts for changes of h, while the tangent frame is held constant. For the former, we require derivatives of T, e.g:

$$\frac{\partial T}{\partial u_2} = \begin{pmatrix} -\langle\partial_u\mathbf{x}_2, \partial_u\mathbf{n}_2\rangle\mathbf{n}_2 - \langle\partial_u\mathbf{x}_2, \mathbf{n}_2\rangle\partial_u\mathbf{n}_2 \\ -\langle\partial_v\mathbf{x}_2, \partial_u\mathbf{n}_2\rangle\mathbf{n}_2 - \langle\partial_v\mathbf{x}_2, \mathbf{n}_2\rangle\partial_u\mathbf{n}_2 \end{pmatrix}, \quad \frac{\partial T}{\partial v_2} = \begin{pmatrix} -\langle\partial_u\mathbf{x}_2, \partial_v\mathbf{n}_2\rangle\mathbf{n}_2 - \langle\partial_u\mathbf{x}_2, \mathbf{n}_2\rangle\partial_v\mathbf{n}_2 \\ -\langle\partial_v\mathbf{x}_2, \partial_v\mathbf{n}_2\rangle\mathbf{n}_2 - \langle\partial_v\mathbf{x}_2, \mathbf{n}_2\rangle\partial_v\mathbf{n}_2 \end{pmatrix}$$

and for the second term we can repeatedly apply the following vector calculus identity:

$$\frac{\partial}{\partial t}\frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|} = \frac{1}{\|\mathbf{z}(t)\|}\frac{\partial \mathbf{z}(t)}{\partial t} - \frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|^3}\left\langle \mathbf{z}(t), \frac{\partial \mathbf{z}(t)}{\partial t}\right\rangle$$

where $\mathbf{z}$ is a vector-valued function that depends on t. The resulting expression is quite messy and thus we only provide numerical values here. In particular, ∇C is given by

$$\nabla C = \begin{bmatrix} -\frac{1}{4} & 0 & -\frac{3}{2} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & 0 & -1 & 0 & \frac{1}{2} \end{bmatrix}$$

and therefore

$$T_{\mathcal{S}}(\bar{\mathbf{x}}) = -A^{-1}\begin{bmatrix} B_1 & B_3 \end{bmatrix} = \begin{pmatrix} -\frac{1}{6} & 0 & \frac{1}{6} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}.$$

Since the parameterizations are orthonormal, the geometric term between $\mathbf{x}_1$ and $\mathbf{x}_3$ is simply

$$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_3) = \left|P_2 A^{-1} B_3\right| G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \begin{vmatrix} \frac{1}{6} & 0 \\ 0 & \frac{1}{2} \end{vmatrix} G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \frac{1}{48}.$$

For a C++ implementation that computes ∇C, please refer to the appendix A2.

4 Walking on the specular manifold

Our rendering algorithms use MCMC to explore the manifold of specular paths, and for this they require some form of local parameterization. With the differential geometry of the specular manifold in hand, we are now able to develop this extremely useful building block. We propose an algorithm that moves one of the endpoints of a specular chain and takes all the intermediate vertices to a valid new configuration. Later, in Section 5, we show how to apply this algorithm to render images.

To simplify the discussion, we will focus on the case where the position of a vertex $\mathbf{x}_n$ of a specular chain $\mathbf{x}_1, \ldots \mathbf{x}_n$ is adjusted to a given new position $\mathbf{x}'_n$, while $\mathbf{x}_1$ is held fixed. We shall also briefly introduce the assumption that $\mathbf{x}_n$ is located on planar surface of infinite extent.

Our manifold walking algorithm is based on two key insights:

1. The A and B matrices (Section 3.3) may be used to map an infinitesimal in-plane movement of $\mathbf{x}_n$ to displacements of the vertices $\mathbf{x}_2, \ldots, \mathbf{x}_{n-1}$. We can use these displacements to approximate a finite change to the path simply by adding an offset to each vertex, but this will move the path off the specular manifold.

2. Ray tracing provides a deterministic means of projecting an off-specular path back onto the space of valid configurations based on its first two vertices. Given $\mathbf{x}_1$ and $\mathbf{x}_2$, we can trace a sequence of rays $\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}$, at each step performing a specular reflection or refraction, and this leads to corrected positions $\mathbf{x}_2^+ \ldots \mathbf{x}_n^+$.

By combining 1. and 2., we obtain a predictor-corrector type algorithm (Figure 5) that performs a step according to a local linear model, followed by a projection that restores the specular configuration, resulting in a new path $\mathbf{x}_1, \mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$, and repeats until convergence. As long as

the prediction step solves the linear model and moves in the tangent space to the manifold, this iteration behaves like Newton's method, exhibiting quadratic convergence near the solution. As with all Newton-like iterations, it is not guaranteed to converge when started far from the solution, since the linear model may not be accurate enough to make progress. But since the model is first-order accurate, the algorithm is *guaranteed* to make forward progress when the constraint function is differentiable and the partial steps are small enough. Our algorithm uses a simple heuristic to decrease the step size when progress is not made, then increase back to full steps to get quadratic convergence as it approaches the target configuration. This iteration is illustrated in Figure 5 and laid out in the following algorithm:

WalkManifold$\big(\mathbf{x}_1, \ldots, \mathbf{x}_n \rightsquigarrow \mathbf{x}'_n\big)$

1 Set $i = 0$ and $\beta = 1$
2 **while** $\|\mathbf{x}_n - \mathbf{x}'_n\| > \varepsilon L$
3 $\quad \mathbf{p} = \mathbf{x}_2 - \beta\, T(\mathbf{x}_2) P_2 A^{-1} B_k T(\mathbf{x}_n)^T (\mathbf{x}'_n - \mathbf{x}_n)$
4 $\quad$ Propagate the ray $\mathbf{x}_1 \rightarrow \mathbf{p}$ through all specular interactions, producing $\mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$.
5 $\quad$ **if** step 4 succeeded and $\|\mathbf{x}_n^+ - \mathbf{x}'_n\| < \|\mathbf{x}_n - \mathbf{x}'_n\|$
6 $\quad\quad \mathbf{x}_2, \ldots, \mathbf{x}_n = \mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$
7 $\quad\quad \beta = \min\{1, 2\,\beta\}$
8 $\quad$ **else**
9 $\quad\quad \beta = \frac{1}{2}\beta$
10 $\quad$ Set $i = i + 1$, and fail if $i > N$.
11 **return** $\mathbf{x}_2, \ldots, \mathbf{x}_{n-1}$

Here, i records the number of iterations until a specified maximum N is reached, and ε is a relative error threshold to a scene-scale length L (we use $L = \max_i \|\mathbf{x}_i\|$, $N = 20$ and $\varepsilon = 10^{-7}$). The variable β denotes a step size that is dynamically adjusted to ensure that steps reduce the target distance. The pseudocode assumes that the plane vertex $\mathbf{x}_n$ has an orthonormal parameterization.

To make this algorithm usable for general specular chains, we must remove the previous assumption that the endpoint $\mathbf{x}_n$ is located on a plane. However, the actual locations of $\mathbf{x}_n^+$ along the way from $\mathbf{x}_n$ to $\mathbf{x}'_n$ are not needed; all that matters is that $\mathbf{x}'_n$ is on the surface at convergence. Therefore, we construct a plane containing $\mathbf{x}_n$ and $\mathbf{x}'_n$, and in step 4 of WalkManifold the last propagation step computes an intersection against this hypothetical plane, ignoring the actual scene geometry. Once the algorithm converges, we must ensure that $\mathbf{x}_{n-1}$ and $\mathbf{x}_n$ are mutually visible before reporting the manifold walk as successful.

In our implementation, we assign the plane normal using the following symmetric orthogonalization procedure

$$\mathbf{n}_{\text{plane}} \coloneqq \gamma\big(\gamma(\mathbf{n} + \mathbf{n}') - \overrightarrow{\mathbf{x}_n\mathbf{x}'_n}\langle\gamma(\mathbf{n} + \mathbf{n}'), \overrightarrow{\mathbf{x}_n\mathbf{x}'_n}\rangle\big)$$

where $\mathbf{n}$ and $\mathbf{n}'$ are the surface normals at $\mathbf{x}_n$ and $\mathbf{x}'_n$, respectively, and $\gamma(\mathbf{v}) \coloneqq \mathbf{v}/\|\mathbf{v}\|$. This ensures that the same plane is used for walks $\mathbf{x}_n \rightsquigarrow \mathbf{x}'_n$ and $\mathbf{x}'_n \rightsquigarrow \mathbf{x}_n$.

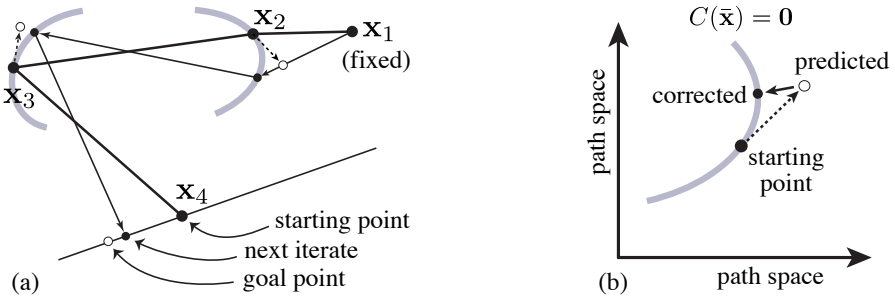


Figure 5: One iteration of updating a path using the specular manifold. **(a)** The path vertices are modified according to a local linear model, which **(b)** corresponds to a step along the tangent plane to the manifold, then **(a)** a nearby valid specular path is found, which **(b)** corresponds to projecting back onto the manifold.

To compute the matrix A, we derived symbolic expressions for the manifold constraints in C (Equation 8). A C++ implementation is given in the appendix. When solving the resulting linear system in step 3, it is beneficial to exploit the special structure of this matrix, which becomes important when processing specular chains with more than about ten vertices. We solve for $\mathbf{p}$ using a block tridiagonal LU factorization, and this reduces the time complexity from $O(n^3)$ to $O(n)$, n being the number of vertices in the chain.

There are several situations under which this algorithm may fail to converge: first, a specular path between $\mathbf{x}_1$ and $\mathbf{x}'_n$ need not exist at all. Secondly, WalkManifold usually cannot find paths that lie on a different connected component of the manifold. Thirdly, when the local structure of the manifold is complex (e.g. due to high-frequency geometric detail of the reflectors and refractors) and $\|\mathbf{x}_n - \mathbf{x}'_n\|$ is large, the iteration may not converge to a solution. Finally, the linear system may not be invertible, which happens when the last path lies at the fold of a wavefront, e.g. a caustic receiving an infinite power per unit area. During the $\sim 10^{12}$ manifold walks performed to produce the results of the paper, this last case only occurred $\sim 1.7 \cdot 10^5$ times, hence it does not appear to be an issue in practice. However, in the MCMC context it is not an problem for the iteration to fail occasionally, as explained in the next section.

The manifold walking algorithm works reliably for large chains with over 10 vertices, especially when it is used by the transition rule discussed in the next section, which only moves the endpoints of chains by a small amount. In our scenes, we observe between 92 and 98% successful walks, each taking 2-3 iterations on average to converge to the tolerance $\varepsilon = 10^{-7}$. The failing 2-8% mainly contain cases where WalkManifold failed for good reasons, because it was asked to walk to a point for which there is no valid configuration on the manifold.

5 Manifold exploration for surfaces

In this section, we present a new transition rule that proposes steps in path space using manifold walks. In the context of MCMC, a transition rule, or *perturbation*, is a random process that generates a *proposal* state conditioned on the current state of a Markov chain. It provides the basic means of navigating through the state space, but to do this correctly the rule must satisfy two basic criteria: transitions must be *reversible* (i.e. return to the previous state with nonzero probability density), and the rule must also supply a function that, up to constant factors, computes the

probability of proposals conditioned on the current state.

To create an efficient sampling procedure, a transition rule should furthermore propose modifications of an appropriate scale. A rule that takes large steps will tend to leave local maxima of the target distribution π, and such steps are rejected with high probability. A rule that takes tiny steps will find most of them accepted, but it will not explore the state space well. Our perturbation is designed so that its scale naturally adapts to the scene, including the geometry and material properties.

The new perturbation supports general scenes and can be used both with the ERPT algorithm by Cline et al. and the path space MLT framework proposed by Veach and Guibas, where it replaces and generalizes the lens, caustic, and multi-chain perturbations. Depending on which combination is used, we call the resulting algorithm either *Manifold Exploration Path Tracing* (MEPT) or *Manifold Exploration Metropolis Light Transport* (MEMLT).

5.1 Manifold perturbation

Given an input path, the manifold perturbation finds a nearby path using a sequence of steps that can be grouped into *sampling* and *connection* phases (Figure 6). The sampling phase chooses a subpath to be modified that consists of three non-specular vertices which are potentially separated by specular chains. We shall denote these non-specular vertices as $\mathbf{x}_a, \mathbf{x}_b$, and $\mathbf{x}_c$. After establishing the type of perturbation to be performed, the sampling step generates a perturbed outgoing direction at the vertex $\mathbf{x}_a$ and propagates it through the specular chain between $\mathbf{x}_a$ and $\mathbf{x}_b$ (if any) until arriving at a new non-specular vertex $\mathbf{x}'_b$ in the neighborhood of $\mathbf{x}_b$. If the path configuration (i.e. the arrangement of specular and nonspecular vertices) changed, the perturbation is rejected immediately.

Up to this point, the proposed scheme is similar to the set of perturbations proposed by Veach and Guibas. However, recall that in their work, perturbations must propagate through the path until arriving at a *pair* of adjacent non-specular vertices ("DD" in Heckbert's [1990] notation) that can be used to establish a connection edge. Any attempt to connect two sampled subpaths that involves a specular vertex must fail, since the probability of creating a valid path in this manner is zero.

In comparison, our perturbation can stop at vertex $\mathbf{x}_b$ and use the WalkManifold algorithm to update the configuration of the specular chain between $\mathbf{x}_b$ and $\mathbf{x}_c$ (Figure 7). This seemingly subtle difference has major repercussions on the types of scenes that can be rendered efficiently. In particular, the resulting method can systematically explore large classes of specular paths instead of having to rely on finding them by chance.

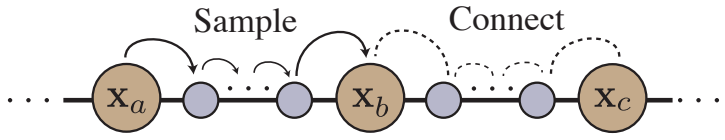


Figure 6: The manifold perturbation samples a perturbed outgoing direction from a vertex $\mathbf{x}_a$ and propagates it through a specular chain (if any) using ray tracing until arriving at a non-specular vertex $\mathbf{x}_b$. To connect the vertices $\mathbf{x}_b$ and $\mathbf{x}_c$, the perturbation performs a manifold walk to determine the positions of intermediate specular vertices (if any).

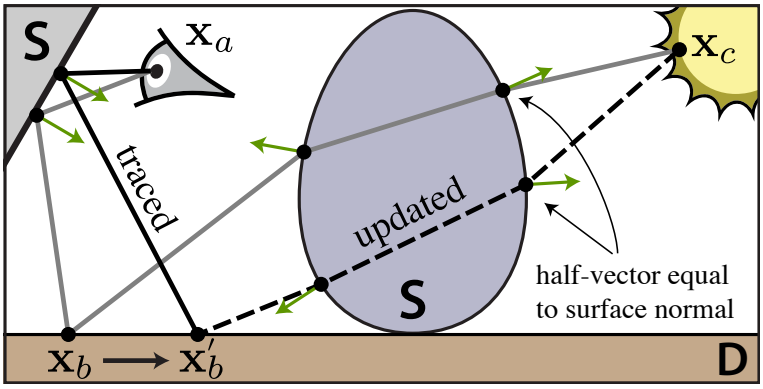


Figure 7: Manifold perturbation example: a slightly perturbed outgoing direction at $\mathbf{x}_a$ is propagated until encountering the non-specular vertex $\mathbf{x}'_b$. Previously, it was not clear how to "connect" $\mathbf{x}'_b$ to $\mathbf{x}_c$ through multiple specular interactions. Our method can find this connection given knowledge about the previous path.

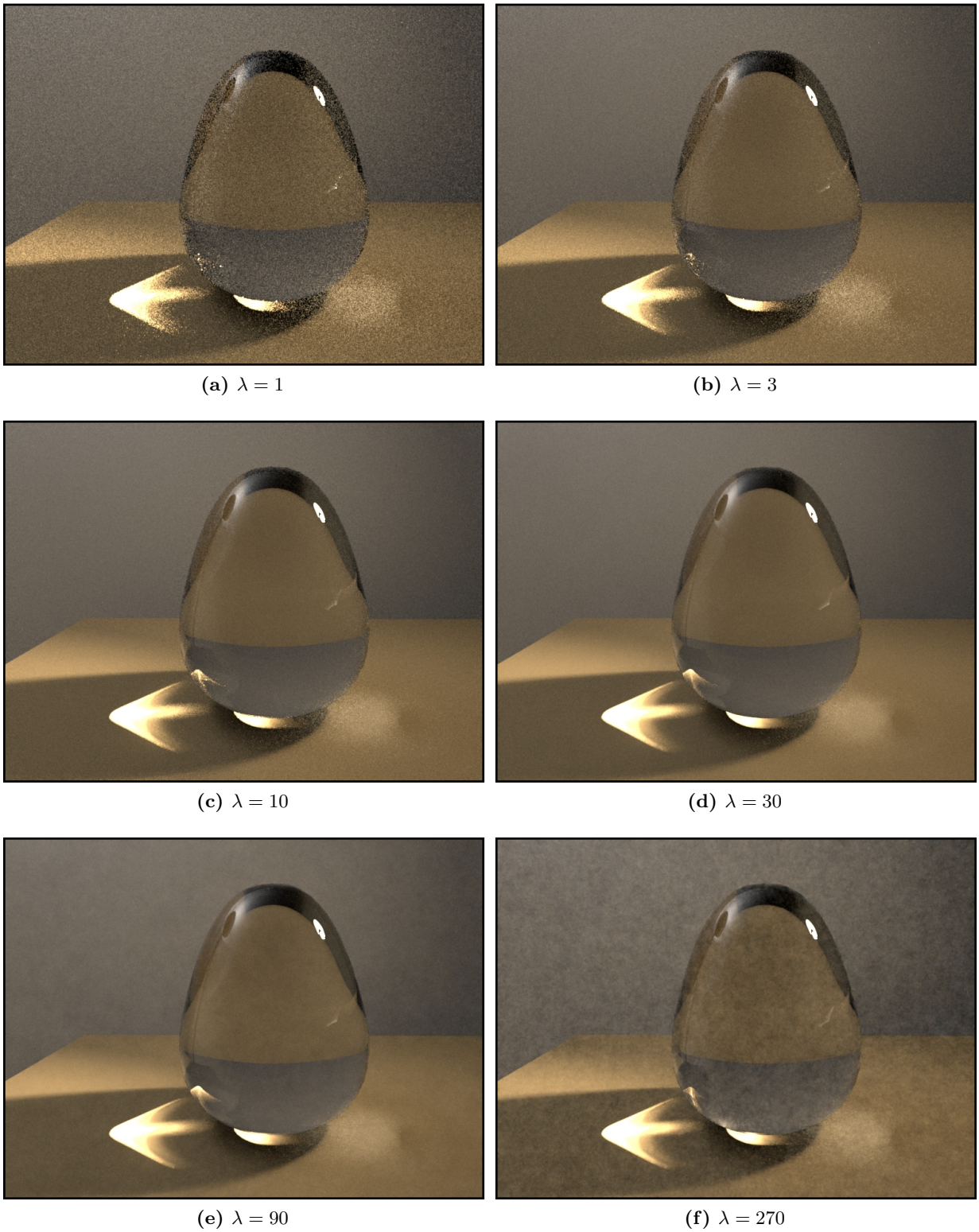
In the following, we will discuss the perturbation in more detail; first for the ideally specular case and then in a more general form that extends to rough surfaces.

Strategy sampling: Motivated by the desire to attempt a large range of different types of path modifications with high probability, the sampling step first chooses among the possible *perturbation strategies* for a given path by selecting three vertices as follows. Given a path $\mathbf{x}_1, \ldots, \mathbf{x}_k$, uniformly select a non-specular initial vertex $\mathbf{x}_a$, as well as a perturbation direction (i.e. towards the light source or towards the camera). Walk along the path in this direction until the first non-specular vertex is encountered, and continue until a second non-specular vertex is found. This path traversal may fail by walking past the end of the path, in which case the strategy sampling phase is simply restarted from scratch. This determines $\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$. For notational convenience, assume that $a < b < c$.

Perturbation sampling: With the overall strategy established, the sampling phase now perturbs the path segment $\mathbf{x}_{a+1}, \ldots, \mathbf{x}_b$. The goal here is to produce a new subpath $\mathbf{x}'_{a+1}, \ldots, \mathbf{x}'_b$ that is "nearby". When the vertex $\mathbf{x}_a$ denotes a surface scattering event with incident and exitant directions $\omega_i = \overrightarrow{\mathbf{x}_a\mathbf{x}_{a-1}}$ and $\omega_o = \overrightarrow{\mathbf{x}_a\mathbf{x}_{a+1}}$, the perturbation determines $\mathbf{x}'_{a+1}$ by tracing a ray in a direction ω'_o that is sampled from a suitable spherical distribution $D(\omega'_o)$ concentrated around ω_o. It is absolutely critical that this distribution generates direction changes of the appropriate scale: for instance, when $\mathbf{x}_a$ is a diffuse material, relative large perturbations are in order. On the other hand, when $\mathbf{x}_a$ is a glossy material that only reflects into a small cone of directions, large perturbations will almost always be rejected, reducing performance.

Observe that a useful hint about the right scale can be obtained directly from the scattering model at $\mathbf{x}_a$, in particular from the associated importance sampling density $p(\omega_i \to \omega_o)$. When this sampling density is high, ω_o is likely located on a sharp peak of the scattering function, and small steps are appropriate.

Our strategy is to sample from a distribution $D(\omega'_o)$ centered at ω_o whose concentration is set so that $D(\omega_o)$ equals $\lambda^2 p(\omega_i \to \omega_o)$. For $D(\omega'_o)$, we use the spherical von Mises-Fisher distribution.



(a) $\lambda = 1$ **(b)** $\lambda = 3$

(c) $\lambda = 10$ **(d)** $\lambda = 30$

(e) $\lambda = 90$ **(f)** $\lambda = 270$

Figure 8: The effects of the λ parameter. This scene was rendered with relatively short Markov Chains (100 steps), hence the range of "good" parameter values is lower than in our other examples scenes. Modeled after a scene by Eric Veach.

The parameter λ (generally set between 50 and 500) specifies how large the perturbations are relative to standard BSDF sampling. This is the main parameter of our technique, and it affects how far perturbations will move in path space. When λ is set to an inappropriately low or high value, the amount of noise present in the output renderings increases. In the first case, too few mutations are accepted, causing the chain to become "stuck" in certain paths for many iterations. In the latter case, the steps taken by the chain are too small to effectively explore path space, and this results in the typical coherent noise patterns that are known from other MLT-type algorithms. A comparison involving different settings is shown in Figure 8. We currently set this parameter manually to achieve a desired acceptance ratio, but this could in theory be automated using adaptive MCMC.

When $\mathbf{x}_a$ is a camera or light source, we choose a new outgoing direction in much the same way, but query the underlying model for the directional density of the associated sampling method (e.g. uniformly choosing pixels in screen-space). To further enlarge the space of possible perturbations, following Veach, we split camera and light source endpoints into two separate vertices corresponding to the position and direction components. When $\mathbf{x}_a$ is such a position vertex, we perturb its location on the aperture or light source by sampling a tangential displacement from a 2D normal distribution with variance $\rho/(2\pi\lambda^2)$, where ρ is the surface area.

After $\mathbf{x}'_{a+1}$ has been determined in this manner, the perturbation is propagated through the specular chain until reaching $\mathbf{x}'_b$. This process is deterministic.

Connection: When there is no specular chain between $\mathbf{x}'_b$ and $\mathbf{x}_c$, the connection step only entails checking that the vertices are mutually visible, and that their scattering models carry illumination along the connection edge. When there is a manifold, we first set

$$\mathbf{x}'_{c-1}, \ldots \mathbf{x}'_{b+1} = \textsc{WalkManifold}(\mathbf{x}_c, \ldots \mathbf{x}_b \rightarrow \mathbf{x}'_b)$$

and then perform the same verification.

Recall that a key requirement of the Metropolis-Hastings algorithm discussed in Section 2.2 was that a nonzero transition probability $T(\mathbf{x}, \mathbf{x}') > 0$ also implies that $T(\mathbf{x}', \mathbf{x}) > 0$. This creates a potential issue when walking on the manifold, because WalkManifold can be *non-reversible*. It might succeed in moving from $\mathbf{x}$ to $\mathbf{x}'$ but fail to move from $\mathbf{x}'$ to $\mathbf{x}$. Even when the reverse iteration converges, the manifold can contain bifurcations so that it may converge to a different solution. Therefore, we always perform another manifold walk in the reverse direction and reject the perturbation if the path did not return to its original configuration. In the example scenes, we observed between 0% and 0.4% non-reversible walks.

Transition probability computation Finally, the change in the contribution function is computed and, together with the transition probabilities, used to randomly accept or reject the proposal with probability

$$r(\bar{x}, \bar{x}') = \min \left\{ 1, \frac{f_j(\bar{\mathbf{x}}')T(\bar{\mathbf{x}}', \bar{\mathbf{x}})}{f_j(\bar{\mathbf{x}})T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')} \right\} \tag{10}$$

where f_j is the contribution function (Equation 4) and $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$, denote the original and proposal path respectively. Note that many factors cancel in the above ratio, particularly all of those in f_j that are associated with the unchanged path segment, or common terms in the transition

probability. For instance, the probability of choosing a particular sampling strategy cancels, since it only depends on the (unchanged) path configuration.

We require that $T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$ and $T(\bar{\mathbf{x}}', \bar{\mathbf{x}})$ express the density of forward and reverse proposals in a common measure. Observe that sampling an outgoing direction ω_o' from $D(\omega_o')$ at $\mathbf{x}_a$, and propagating it through the first specular chain, produces area density $D^{\perp}(\omega_o')\, G(\mathbf{x}_a \leftrightarrow \cdots \leftrightarrow \mathbf{x}_b')$ on the surface at $\mathbf{x}_b'$ (where $D^{\perp}(\omega_o') = D(\omega_o')/|\cos(\mathbf{n}_a, \omega_o)|$ denotes probability with respect to the projected solid angle measure at $\mathbf{x}_a$). This is the needed transition probability $T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$.

In a MCMC-based rendering system, we will generally want to sample paths based on their contribution to the *entire* image rather than to a single pixel. This is accomplished by replacing $W_e^{(j)}$ in (4) with an importance function that measures the overall luminance received on the image plane.

5.2 Extension to glossy materials

The method just presented can be used for scenes with both specular and non-specular transport, and the two classes are treated separately. This is unfortunate, since a near-specular chain through an almost-smooth dielectric object cannot be explored as effectively as a perfectly specular one. However, it turns out that a simple generalization suffices to encompass these materials as well.

The path space integrand corresponding to a chain of glossy interactions has its energy concentrated in a thin "band" *near* the specular manifold, and the key idea of how we handle glossy materials is to take steps along a family of *offset manifolds* that are parallel to the specular manifold, so that path space near the specular manifold can be explored without stepping out of this thin band of near-specular transport. In this section, we add a simple extension that endows the perturbation with the ability to walk on offset manifolds and to recognize when this is appropriate.

For this, we first replace Equation (8) with the *offset manifold*

$$\mathcal{S}_{\mathbf{o}} = \{\bar{\mathbf{x}} \mid C(\bar{\mathbf{x}}) = \mathbf{o}\}\,, \tag{11}$$

where $\mathbf{o}$ captures the offset from ideal specular transport. Inspecting C (Equation 6) yields an intuitive explanation for the contents of the vector $\mathbf{o}$ in terms of microfacet theory. The two entries associated with each vertex $\mathbf{x}_i$ record the microsurface normal $\mathbf{m}_i$ responsible for the reflection or refraction (which is now different from the shading normal $\mathbf{n}_i$), projected into the shading tangent space $T(\mathbf{x}_i)$. Our extended perturbation then preserves the projection of this microgeometry normal $\mathbf{m}_i$ as an invariant during the manifold traversal.

Changes to the manifold perturbation: Since the differential geometry of the offset manifold is identical to that of the ordinary manifold, the only required change in WalkManifold affects the ray tracing step on line four, where the algorithm now reflects and refracts using $\mathbf{m}_i$ instead of $\mathbf{n}_i$. Similarly, the deterministic phase of the manifold perturbation responsible for propagating the sampled direction at $\mathbf{x}_a$ to a position $\mathbf{x}_b'$ uses these normals instead. Note that it is straightforward to handle both cases, near-specular and specular perturbations, using the same implementation.

Recognizing near-specular transport: An important aspect about the treatment of general scattering is the decision of whether the surface associated with a scattering event is "smooth

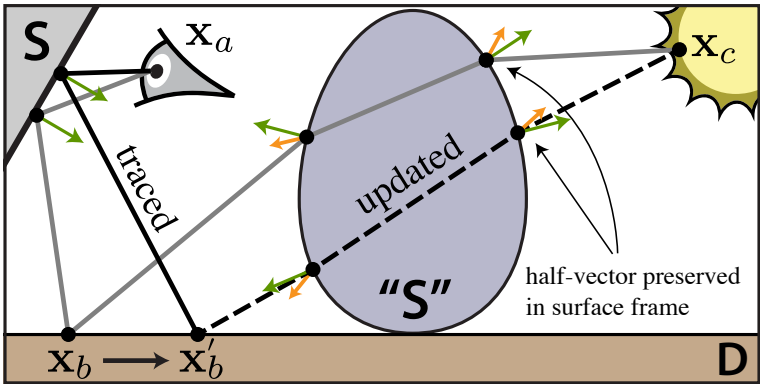
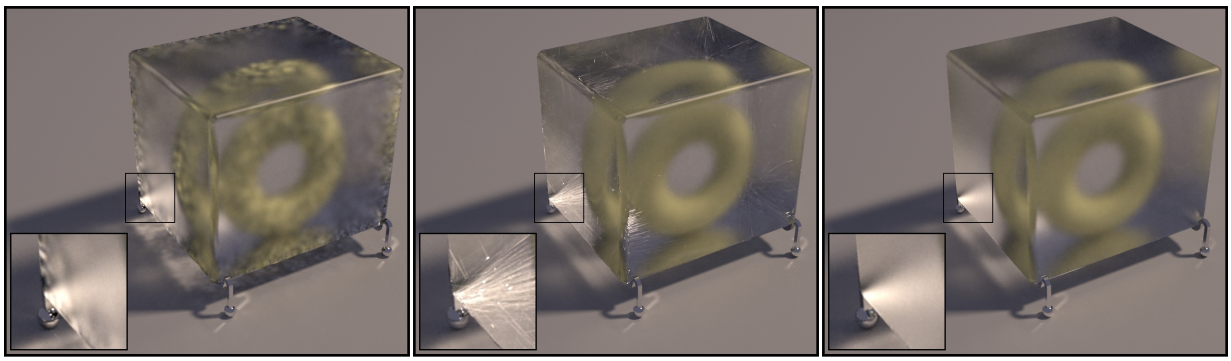


Figure 9: Perturbation of path with near-specular surface interactions: instead of requiring that the half-vectors agree with the surface normals, their direction is preserved in the surface frame.

enough" to be classified as part of a specular chain. We make this decision randomly by assigning a specular probability $\psi(\mathbf{x}_i)$ to each vertex that takes on values 0 and 1 when $\mathbf{x}_i$ is diffuse and specular, respectively, and values in $(0, 1)$ when $\mathbf{x}_i$ is at a rough interface. This avoids the issues of "hard" classifications that are commonly used in rendering algorithms. For specifics on our specular probability function ψ, please refer to the appendix.

Transition probability: In the purely specular case discussed earlier, the proposal and target distributions were both supported on the same space $\mathcal{S}$. This property permitted computing transition probabilities under an arbitrary projection (e.g. onto the vertex $\mathbf{x}_b$), since the determinant of the associated change of variables canceled when considering ratios of the form f_j/T.

In the glossy case, this does not hold anymore: T is a distribution on an offset manifold $\mathcal{S}_\mathbf{o}$, whereas f_j is defined on the higher-dimensional space $\bigcup_\mathbf{o} \mathcal{S}_\mathbf{o}$. To compute transition probabilities, we must perform a change of variables to separate out these "perpendicular" dimensions in f_j,



(a) always non-specular **(b)** always specular **(c)** probabilistic

Figure 10: Consistently classifying glossy materials as non-specular or specular produces unsatisfactory results. Instead, our method makes this decision randomly whenever encountering a rough object (modeled after a scene by Cline et al.)

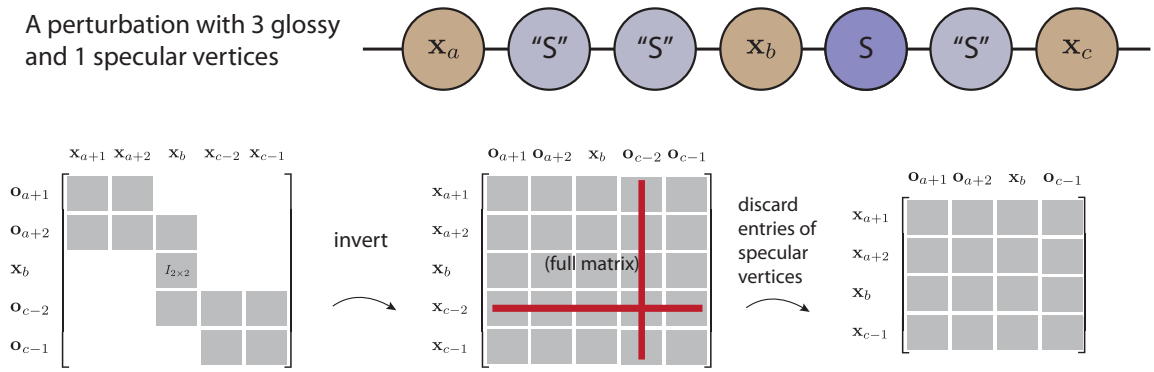


Figure 11: An illustration of the matrices involved in the glossy transition probability computation.

and this causes a determinant to appear in the final transition probability:

$$T(\bar{\mathbf{x}}, \bar{\mathbf{x}}') = T_{\text{spec}}(\bar{\mathbf{x}}, \bar{\mathbf{x}}') \left| \frac{\partial \left[\mathbf{x}_b, \mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\right]}{\partial \left[\mathbf{x}_b, \mathbf{o}_{i_1}, \ldots, \mathbf{o}_{i_k}\right]} \right| (1 - \psi(\mathbf{x}_b))(1 - \psi(\mathbf{x}_c))) \prod_{i=a+1, i\neq b}^{c-1} \psi(\mathbf{x}_i). \tag{12}$$

Here, T_{spec} is the transition probability of the purely specular case and the indices $i_1, \ldots, i_k \in \{a+1, \ldots, c-1\} \setminus \{b\}$ refer to glossy vertices that were classified as specular. The terms involving ψ represent the discrete probability of performing this classification; roughness values may change during a perturbation, and hence we must account for them to maintain detailed balance.

The determinant in (12) is not hard to compute. Recall that the A-matrix (Section 3.3) maps perturbations of the vertex positions to changes of the half-vectors. Here, we seek the opposite: how the non-specular vertices move as a function of $\mathbf{o}$ and $\mathbf{x}_b$. We therefore compute the matrix A over the vertex range $\mathbf{x}_{a+1}, \ldots, \mathbf{x}_{c-1}$ with one small adjustment: the two rows associated with vertex $\mathbf{x}_b$ are set so that there is a 2×2 identity matrix on the diagonal and zeroes everywhere else. (Figure 11) We invert this matrix and discard all rows and columns in this inverse that are associated with specular vertices, and the determinant of the result is the sought change of variables factor. Note that when the two chains only consists of glossy vertices, it is equivalent (and considerably faster) to compute the inverse determinant of the block tridiagonal matrix.

6 A unified path space framework for surfaces and volumes

We have presented Manifold Exploration in the context of surfaces, but the extension to volumes poses no fundamental difficulties. Before we can proceed, however, we require a description of how volumetric interactions are modeled in the path-space setting.

The purpose of this section is to find an operator-based description of surface and volume light transport similar in style to the frameworks presented by Arvo [1995] and Veach [1997], which is later used to derive a path-space integral. Such a unified surface and volume path space has been proposed in prior work by Pauly et al. [2000], but the context of that work was computational, and hence it only definitions were supplied.

The goal here is to provide a complete derivation that provides solid theoretical underpinnings

for the development of new path-space rendering algorithms. We roughly follow the approach of Chapter 4 in Eric Veach's Ph.D. thesis [Veach 1997] and conclude with an extended set of operators that can describe interactions with surfaces and volumes.

6.0.1 Definitions

It will be convenient to establish some notation: as before, $\mathcal{M}$ denotes the set of surfaces. We define the distance to the next surface as seen by the ray $(\mathbf{x}, \omega) \subseteq \mathbb{R}^3 \times S^2$ as

$$d_{\mathcal{M}}(\mathbf{x}, \omega) \coloneqq \inf \{d > 0 \mid \mathbf{x} + d\omega \in \mathcal{M}\}$$

where $\inf \emptyset = \infty$. Based on this distance, we can define a *ray-casting function* $\mathbf{x}_{\mathcal{M}}$:

$$\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega) = \mathbf{x} + d_{\mathcal{M}}(\mathbf{x}, \omega)\omega.$$

We shall also make use of the following notation to express integration along rays

$$\int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{z})\, \mathrm{d}\mathbf{z} \coloneqq \int_0^{d_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{x} + t\omega)\, \mathrm{d}t.$$

Note that that while $\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega)$ is generally undefined for "points at infinity" ($d_{\mathcal{M}}(x, \omega) = \infty$), it has a well-defined meaning when encountered in integrals of the above form. We will also write

$$\int_{\mathbf{x}}^{\mathbf{y}} f(\mathbf{z})\, \mathrm{d}\mathbf{z} \coloneqq \int_0^{\|\mathbf{x}-\mathbf{y}\|} f(\mathbf{x} + t\,\overrightarrow{\mathbf{xy}})\, \mathrm{d}t, \quad \text{where} \quad \overrightarrow{\mathbf{xy}} \coloneqq \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|}.$$

6.1 Light transport model

The *radiative transfer equation* (RTE) is commonly used to describe the steady-state light transport on a three-dimensional domain $\mathcal{V} \subseteq \mathbb{R}^3$. It is given by the following energy balance equation:

$$(\omega \cdot \nabla)\ L(\mathbf{x}, \omega) + \sigma_t(\mathbf{x})\ L(\mathbf{x}, \omega) = \sigma_s(\mathbf{x}) \int_{S^2} f_p(\mathbf{x}, \omega' \to \omega)\ L(\mathbf{x}, \omega')\, \mathrm{d}\sigma_{\mathbf{x}}(\omega') + L_e(\mathbf{x}, \omega), \qquad \mathbf{x} \in \mathcal{V}^\circ. \tag{13}$$

where σ_s and σ_t are the scattering and extinction coefficients, f_p is the phase function, L denotes the volume radiance, and L_e represents emitted radiance. $d\sigma_{\mathbf{x}}$ denotes differential solid angles at $\mathbf{x}$. Please note that due to the volumetric setting, the emission term L_e has units of radiance *per unit length* at interior points $x \in \mathcal{V}^\circ$.

We will later specify boundary conditions on the set of surfaces $\mathcal{M} \subseteq \mathcal{V}$, hence the above equation only holds on the interior of the domain, i.e. $\mathcal{V}^\circ \coloneqq \mathcal{V} \setminus \mathcal{M}$.

Restricted to a line of direction ω parameterized by $s \in \mathbb{R}$, Equation (13) can be rewritten as a one-dimensional ordinary differential equation of the form

$$L'(s) + \sigma_t(s)L(s) = Z(s)$$

where $Z(s)$ represents the emitted and in-scattered radiance at s. Given an initial condition at

$s = 0$, this ODE has the solution

$$\begin{aligned} L(s) &= \exp\left(-\int_0^s \sigma_t(r)\,\mathrm{d}r\right)\left[L(0) + \int_0^s \exp\left(\int_0^t \sigma_t(r)\,\mathrm{d}r\right) Z(t)\,\mathrm{d}t\right] \\ &= \exp\left(-\int_0^s \sigma_t(r)\,\mathrm{d}r\right) L(0) + \int_0^s \exp\left(-\int_t^s \sigma_t(r)\,\mathrm{d}r\right) Z(t)\,\mathrm{d}t \end{aligned} \tag{14}$$

where the first term corresponds to attenuated radiance from surfaces and the second term accounts for in-scattered radiance.

In this solution, the parameterization and boundary condition at $s = 0$ were arbitrarily chosen. In general, the boundary condition will be specified by the nearest surface visible along the ray $(\mathbf{x}, -\omega)$. To formalize this constraint, we will first define the *reduced surface radiance* at $(\mathbf{x}, \omega)$ as

$$L_{\mathrm{red}}(\mathbf{x}, \omega) \coloneqq \begin{cases} L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega), \omega)\, \tau(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega) \leftrightarrow \mathbf{x}), & d_{\mathcal{M}}(\mathbf{x}, -\omega) < \infty \\ 0, & \text{otherwise} \end{cases}$$

where

$$\tau(\mathbf{x} \leftrightarrow \mathbf{y}) \coloneqq \exp\left(-\int_{\mathbf{x}}^{\mathbf{y}} \sigma_t\left(\mathbf{z}\right)\,\mathrm{d}\mathbf{z}\right)$$

and L_o represents the surface radiance scattered into direction ω (we will consider this term in more detail later). Now, we can substitute a suitable parameterization into Equation (14) and replace the first term by the reduced surface radiance, which yields the integral form of the radiative transfer equation:

$$\begin{aligned} L(\mathbf{x}, \omega) = L_{\mathrm{red}}(\mathbf{x}, \omega) + \int_0^{d_{\mathcal{M}}(\mathbf{x}, -\omega)} & \tau(\mathbf{x} \leftrightarrow \mathbf{x} - t\omega) \\ & \left(\sigma_s(\mathbf{x} - t\omega) \int_{S^2} f_p(\mathbf{x} - t\omega, \omega' \to \omega) L(\mathbf{x} - t\omega, \omega')\,\mathrm{d}\sigma_{\mathbf{x}-t\omega}(\omega') + L_e(\mathbf{x} - t\omega, \omega)\right)\mathrm{d}t \end{aligned}$$

which can be written more compactly using the introduced notation:

$$L(\mathbf{x}, \omega) = L_{\mathrm{red}}(\mathbf{x}, \omega) + \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \left(\sigma_s(\mathbf{y}) \int_{S^2} f_p(\mathbf{y}, \omega' \to \omega) L(\mathbf{y}, \omega')\,\mathrm{d}\sigma_{\mathbf{y}}(\omega') + L_e(\mathbf{y}, \omega)\right)\mathrm{d}\mathbf{y}. \tag{15}$$

6.2 Boundary conditions

By specifying boundary conditions on the surfaces $\mathcal{M}$, we will invariably also introduce discontinuities in the volumetric radiance function L. These discontinuities are problematic when referring to the surface radiance field, since it will generally not be identical on both sides.

Analogous to a one-sided limit in the 1D case, we shall therefore distinguish between the front- and back-facing components $L^+(\mathbf{x}, \omega)$ and $L^-(\mathbf{x}, \omega)$ as determined by the surface normal $N(\mathbf{x})$ for $\mathbf{x} \in \mathcal{M}$. Based on this separation, the more intuitive incident and exitant radiance functions

can then be defined as

$$L_i(\mathbf{x},\omega) \coloneqq \begin{cases} L^+(\mathbf{x},-\omega), & \omega \cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x},-\omega), & \omega \cdot N(\mathbf{x}) < 0 \end{cases} \quad \text{and} \quad L_o(\mathbf{x},\omega) \coloneqq \begin{cases} L^+(\mathbf{x},\omega), & \omega \cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x},\omega), & \omega \cdot N(\mathbf{x}) < 0 \end{cases}.$$

With the help of these definitions, we can introduce the usual boundary condition known as the *rendering equation* [Kajiya 1986]:

$$L_o(\mathbf{x},\omega) = \int_{S^2} f_s(\mathbf{x},\omega' \to \omega) L_i(\mathbf{x},\omega') \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega') + L_e(\mathbf{x},\omega), \quad \mathbf{x} \in \mathcal{M} \tag{16}$$

where f_s is the BSDF of the surface.

Away from surfaces, the distinction between L_i and L_o is technically not necessary (involving only a change of direction) but we introduce it here for consistency. Specifically, we will set $L_i(\mathbf{x},\omega) \coloneqq L(\mathbf{x},-\omega)$ and $L_o(\mathbf{x},\omega) \coloneqq L(\mathbf{x},\omega)$ for $\mathbf{x} \in \mathcal{V}^\circ$ and to analogously express the integral form of the RTE (15) in terms of L_i and L_o:

$$L_o(\mathbf{x},\omega) = L_{\mathrm{red}}(\mathbf{x},\omega) + \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},-\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \left(\sigma_s(\mathbf{y}) \int_{S^2} f_p(\mathbf{y},-\omega' \to \omega) L_i(\mathbf{y},\omega') \, \mathrm{d}\sigma_{\mathbf{y}} + L_e(\mathbf{y},\omega) \right) \mathrm{d}\mathbf{y}. \tag{17}$$

Note the negated ω' term in the spherical integral, which has a positive correspondence in the surface boundary condition (16). This apparent discrepancy is due to the the different parameter conventions of surface and volume scattering models.[3]

6.3 Operator notation

Volumetric light transport can be seen as the alternation of two steps: *scattering* on a surface or within the medium, followed by *propagation* and *attenuation*. Analogous to [Veach 1997] and [Arvo 1995], these two steps can be formulated as linear operators defined on the space of radiance functions. The goal of this section is to partition Equations (16) and (17) so that they can be expressed in this manner.

We define the scattering operator $\mathbf{K}$ as one of the in-scattering integrals in Equations (16) or (17) dependent on whether or not $\mathbf{x} \in \mathcal{M}$:

$$(\mathbf{K}h)(\mathbf{x},\omega) \coloneqq \begin{cases} \displaystyle\int_{S^2} f_s(\mathbf{x},\omega' \to \omega) \, h(\mathbf{x},\omega') \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega'), & \mathbf{x} \in \mathcal{M} \\ \sigma_s(\mathbf{x}) \displaystyle\int_{S^2} f_p(\mathbf{x},-\omega' \to \omega) \, h(\mathbf{x},\omega') \, \mathrm{d}\sigma_{\mathbf{x}}(\omega'), & \text{otherwise} \end{cases} \tag{18}$$

On surfaces, $\mathbf{K}$ turns incident radiance into outgoing radiance. On the interior of the domain, it turns incident radiance into outgoing radiance per unit length.

In comparison, the transport operator $\mathbf{G}$ has a single definition on the whole domain. Its role is to transform outgoing radiance per unit length from the volume and outgoing radiance from

[3] As a somewhat unfortunate consequence of notations in different fields, i n volume scattering models, the incident direction arguments points *towards* the scattering location, whereas it points *away* in the surface case.

surfaces into incident radiance.

$$(\mathbf{G}h)(\mathbf{x},\omega) \coloneqq \int_{\mathbf{x}}^{\mathbf{x}_\mathcal{M}(\mathbf{x},\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y})\, h(\mathbf{y},-\omega)\,\mathrm{d}\mathbf{y} + \tau(\mathbf{x} \leftrightarrow \mathbf{x}_\mathcal{M}(\mathbf{x},\omega))\, h(\mathbf{x}_\mathcal{M}(\mathbf{x},\omega),-\omega) \tag{19}$$

This leads to interesting differences in comparison to previous work: to match equations (16) and (17), we arrive at an equilibrium equation of the form

$$L_i = \mathbf{G}\left(\mathbf{K}L_i + L_e\right) \tag{20}$$

whereas $L_o = L_e + \mathbf{K}\mathbf{G}L_o$ was used by [Veach 1997]. The differences arise since volume light transport forces us to deal with both radiance and radiance per unit length. To express the equilibrium condition purely in terms of radiance, the order of the $\mathbf{G}$ and $\mathbf{K}$ operators must thus be reversed. Assuming invertibility, the solution operator $\mathbf{S}$ can now be found:

$$\begin{aligned} & L_i = \mathbf{G}\left(\mathbf{K}L_i + L_e\right) \\ \Leftrightarrow\ & \left(\mathbf{I} - \mathbf{G}\mathbf{K}\right) L_i = \mathbf{G}L_e \\ \Leftrightarrow\ & L_i = \underbrace{\left(\mathbf{I} - \mathbf{G}\mathbf{K}\right)^{-1}\mathbf{G}}_{=:\,\mathbf{S}} L_e. \end{aligned}$$

When the following Neumann series converges in operator norm, the solution operator can also be expressed as

$$\mathbf{S} = \sum_{k=0}^{\infty} \left(\mathbf{G}\mathbf{K}\right)^k \mathbf{G}. \tag{21}$$

6.4 Unified path integral formulation

Having defined the necessary operators, we can now proceed to find the desired path integral formulation. Note that in the following, we will often switch between parameterizing functions in terms of directions and positions. The arrow direction indicates the flow of light, which leads to the following conventions:

$$\begin{aligned} L_e(\mathbf{x} \to \mathbf{y}) &\coloneqq L_e(\mathbf{x}, \overrightarrow{\mathbf{x}\mathbf{y}}) \\ W_e(\mathbf{x} \leftarrow \mathbf{y}) &\coloneqq W_e(\mathbf{x}, \overrightarrow{\mathbf{x}\mathbf{y}}) \\ f_s(\mathbf{x} \to \mathbf{y} \to \mathbf{z}) &\coloneqq f_s(\mathbf{y}, \overrightarrow{\mathbf{y}\mathbf{x}}, \overrightarrow{\mathbf{y}\mathbf{z}}) \\ f_p(\mathbf{x} \to \mathbf{y} \to \mathbf{z}) &\coloneqq f_p(\mathbf{y}, \overrightarrow{\mathbf{x}\mathbf{y}}, \overrightarrow{\mathbf{y}\mathbf{z}}) \end{aligned}$$

Here, W_e denotes emitted importance. When dealing with rays that do not intersect any surfaces, we consider the ray-casting operator $\mathbf{x}_\mathcal{M}(\mathbf{x},\omega)$ to return special 'points at infinity'. This ensures that the following is well-defined even if $d_\mathcal{M}(\mathbf{x},\omega) = \infty$:

$$L_e(\mathbf{x} \to \mathbf{x}_\mathcal{M}(\mathbf{x},\omega)) = L_e(\mathbf{x},\omega).$$

6.4.1 Change of variables

When dealing with light transport on surfaces, it is often convenient to switch between integration over a sphere and integration over all surfaces of the scene, e.g.

$$\int_{S^2} f(\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega))\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega) = \int_{\mathcal{M}} f(\mathbf{y})\,G_{\mathrm{surf}}(\mathbf{x}\leftrightarrow\mathbf{y})\,\mathrm{d}A(\mathbf{y}).$$

This change of variables involves the geometric term

$$G_{\mathrm{surf}}(\mathbf{x}\leftrightarrow\mathbf{y}) = V(\mathbf{x}\leftrightarrow\mathbf{y})\cdot\frac{|\cos\theta_1\cos\theta_2|}{\|\mathbf{x}-\mathbf{y}\|^2}$$

where θ_1 and θ_2 denote the angles that $\overrightarrow{\mathbf{xy}}$ makes with the surface normals $N(\mathbf{x})$ and $N(\mathbf{y})$, respectively, and V is a visibility function defined as

$$V(\mathbf{x}\leftrightarrow\mathbf{y}) \coloneqq \begin{cases} 1, & \text{if } \{\alpha\mathbf{x}+(1-\alpha)\mathbf{y} \mid \alpha\in(0,1)\}\cap\mathcal{M}=\emptyset \\ 0, & \text{otherwise} \end{cases}$$

In the volume setting, a similar change of variables is possible: we consider a volume integral over rays radially emanating from a point $\mathbf{x}$. Fubini's theorem and some manipulation leads to

$$\begin{aligned} &\int_{S^2}\int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{y})\,\mathrm{d}\mathbf{y}\,\mathrm{d}\omega \\ &= \int_{S^2}\int_0^{\infty} f(\mathbf{x}+r\omega)V(\mathbf{x}\leftrightarrow\mathbf{x}+r\omega)\,\mathrm{d}r\,\mathrm{d}\omega \\ &= \int_0^{\infty}\frac{1}{r^2}\int_{S^2(\mathbf{x},r)} f(\mathbf{y})V(\mathbf{x}\leftrightarrow\mathbf{y})\,\mathrm{d}\mathbf{y}\mathrm{d}t \\ &= \int_{\mathbb{R}^3}\frac{V(\mathbf{x}\leftrightarrow\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|^2} f(\mathbf{y})\,\mathrm{d}\mathbf{y}. \end{aligned}$$

To handle all possible combinations of volume and surface endpoints, we redefine the basic geometry term as as follows:

$$G(\mathbf{x}\leftrightarrow\mathbf{y}) \coloneqq V(\mathbf{x}\leftrightarrow\mathbf{y})\cdot\frac{D_{\mathbf{x}}\left(\overrightarrow{\mathbf{xy}}\right)D_{\mathbf{y}}\left(\overrightarrow{\mathbf{yx}}\right)}{\|\mathbf{x}-\mathbf{y}\|^2}$$

where

$$D_{\mathbf{a}}(\omega) \coloneqq \begin{cases} |N(\mathbf{a})\cdot\omega|, & \mathbf{a}\in\mathcal{M} \\ 1, & \text{otherwise} \end{cases}$$

6.4.2 Path space measurement integral

Based on the generalized operators $\mathbf{G}$ and $\mathbf{K}$, we can proceed to find a path integral formulation of light transport similar to what is done in Chapter 8 of Eric Veach's thesis [Veach 1997].

We begin by considering a measurement I_j, which is defined as the ray-space convolution of the

importance $W_e^{(j)}$ emitted by sensor j with the incident radiance L_i (see [Veach 1997] for details):

$$\begin{aligned} I_j &= \langle W_e^{(j)}, L_i \rangle \\ &= \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}). \end{aligned} \tag{22}$$

For simplicity, we restrict the ray origins to the boundary, hence volumetric sensors are not supported. The goal of the path space approach is to express the measurement integral (22) over the set of transport paths $\mathcal{P}$

$$I_j = \int_{\mathcal{P}} f_j(\bar{x}) \, \mathrm{d}\mu(\bar{x}),$$

where μ is a measure on $\mathcal{P}$, and f_j is a contribution weighting function specific to the measurement.

To render the definitions precise, let us define the path space $\mathcal{P}$ as the union of all fixed-length paths formed by concatenating vertices from $\mathcal{M}$ and $\mathcal{V}^\circ$ according to a configuration vector $\mathbf{c} \in \{0,1\}^k$, i.e.

$$\mathcal{P} \coloneqq \bigcup_{k=1}^{\infty} \bigcup_{\mathbf{c} \in \{0,1\}^k} \mathcal{P}_k^{\mathbf{c}}$$

where

$$\mathcal{P}_k^{\mathbf{c}} \coloneqq \{\mathbf{x}_0 \mathbf{x}_1 \cdots \mathbf{x}_k \mid (\mathbf{x}_i \in \mathcal{M} \text{ if } \mathbf{c}_i = 0) \text{ and } (\mathbf{x}_i \in \mathcal{V}^\circ \text{ if } \mathbf{c}_i = 1) \;\; (i = 1, \ldots, k)\}.$$

Using the Lebesgue measures for area and volume on $\mathcal{M}$ and $\mathcal{V}^\circ$, we can define a combined product measure on $\mathcal{P}$:

$$\mu(D) \coloneqq \sum_{k=1}^{\infty} \sum_{\mathbf{c} \in \{0,1\}^k} \mu_k^{\mathbf{c}}(D \cap \mathcal{P}_k^{\mathbf{c}}) \quad \text{where} \quad \mu_k^{\mathbf{c}}(D) \coloneqq \int_D \prod_{i=1}^{k} \begin{cases} \mathrm{d}A(\mathbf{x}_i), & \mathbf{c}_i = 0 \\ \mathrm{d}V(\mathbf{x}_i), & \mathbf{c}_i = 1 \end{cases}$$

To find the path space formulation, let us insert the operator form of the equilibrium equation (20) into the measurement integral (22):

$$I_j = \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega)(\mathbf{G}(\mathbf{K}L_i + L_e))(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}).$$

Expanding the $\mathbf{G}$ operator leads to

$$\begin{aligned} = \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) \Bigg[& \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \, [\mathbf{K}L_i + L_e](\mathbf{y}, -\omega) \, \mathrm{d}\mathbf{y} \\ & + \; \tau(\mathbf{x} \leftrightarrow \mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega)) \, [\mathbf{K}L_i + L_e](\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega), -\omega) \Bigg] \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}). \end{aligned}$$

Using a change of variables (c.f. Section 6.4.1), we can turn the two summands into a volume and a surface integral, respectively:

$$= \int_{\mathcal{M}} \Bigg[\int_{\mathcal{V}^\circ} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y})\, \tau(\mathbf{x} \leftrightarrow \mathbf{y})\, G(\mathbf{x} \leftrightarrow \mathbf{y})\, [\mathbf{K}L_i + L_e](\mathbf{y} \rightarrow \mathbf{x})\, \mathrm{d}V(\mathbf{y}) + \int_{\mathcal{M}} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y}) \tau(\mathbf{x} \leftrightarrow \mathbf{y})\, G(\mathbf{x} \leftrightarrow \mathbf{y}) [\mathbf{K}L_i + L_e](\mathbf{y} \rightarrow \mathbf{x})\, \mathrm{d}A(\mathbf{y}) \Bigg] \mathrm{d}A(\mathbf{x}).$$

Using the fact that $\mathbf{K}L_i = \mathbf{K}\mathbf{S}L_e = \sum_{k=1}^{\infty} (\mathbf{K}\mathbf{G})^k L_e$, this can be rewritten as

$$= \sum_{k=0}^{\infty} \int_{\mathcal{M}} \Bigg[\int_{\mathcal{V}^\circ} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y})\, \bar{G}(\mathbf{x} \leftrightarrow \mathbf{y})\, [(\mathbf{K}\mathbf{G})^k L_e](\mathbf{y} \rightarrow \mathbf{x})\, \mathrm{d}V(\mathbf{y}) + \int_{\mathcal{M}} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y})\, \bar{G}(\mathbf{x} \leftrightarrow \mathbf{y}) [(\mathbf{K}\mathbf{G})^k L_e](\mathbf{y} \rightarrow \mathbf{x})\, \mathrm{d}A(\mathbf{y}) \Bigg] \mathrm{d}A(\mathbf{x}). \tag{23}$$

where we have defined the *attenuated geometric term* $\bar{G}(\mathbf{x} \leftrightarrow \mathbf{y}) \coloneqq G(\mathbf{x} \leftrightarrow \mathbf{y})\tau(\mathbf{x} \leftrightarrow \mathbf{y})$.

In the above geometric sum, the operators $\mathbf{G}$ and $\mathbf{K}$ are reversed in comparison previous encounters (e.g. Equation (21)). This makes it possible to perform additional simplifications: consider the concatenated operator $\mathbf{K}\mathbf{G}$ for $\mathbf{y} \in \mathcal{M}$:

$$(\mathbf{K}\mathbf{G}h)(\mathbf{y} \rightarrow \mathbf{x}) = \int_{S^2} f_s\left(\mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega') \rightarrow \mathbf{y} \rightarrow \mathbf{x}\right) \Bigg[\int_{\mathbf{y}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{y},\omega')} \tau(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z}, -\omega')\, \mathrm{d}\mathbf{z} + \tau(\mathbf{y} \leftrightarrow \mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega'))\, h(\mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega'), -\omega') \Bigg] \mathrm{d}\sigma_{\mathbf{y}}^{\perp}(\omega').$$

As before, we can perform a change of variables to volume and surface integrals:

$$= \int_{\mathcal{V}^\circ} f_s\left(\mathbf{z} \rightarrow \mathbf{y} \rightarrow \mathbf{x}\right) \bar{G}(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \rightarrow \mathbf{y})\, \mathrm{d}V(\mathbf{z}) + \int_{\mathcal{M}} f_s\left(\mathbf{z} \rightarrow \mathbf{y} \rightarrow \mathbf{x}\right) \bar{G}(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \rightarrow \mathbf{y})\, \mathrm{d}A(\mathbf{z})$$

For $\mathbf{y} \in \mathcal{V}^\circ$, we get

$$(\mathbf{K}\mathbf{G}h)(\mathbf{y} \rightarrow \mathbf{x}) = \sigma_s \Bigg[\int_{\mathcal{V}^\circ} f_p\left(\mathbf{z} \rightarrow \mathbf{y} \rightarrow \mathbf{x}\right) \bar{G}(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \rightarrow \mathbf{y})\, \mathrm{d}V(\mathbf{z}) + \int_{\mathcal{M}} f_p\left(\mathbf{z} \rightarrow \mathbf{y} \rightarrow \mathbf{x}\right) \bar{G}(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \rightarrow \mathbf{y})\, \mathrm{d}A(\mathbf{z}) \Bigg]$$

To unify the two cases, we define the generalized scattering function $\bar{f}$ as

$$\bar{f}(\mathbf{z}\to\mathbf{y}\to\mathbf{x}) \coloneqq \begin{cases} \sigma_s f_p(\mathbf{z}\to\mathbf{y}\to\mathbf{x}), & \mathbf{y}\in\mathcal{V}^\circ \\ f_s(\mathbf{z}\to\mathbf{y}\to\mathbf{x}), & \mathbf{y}\in\mathcal{M} \end{cases} \tag{24}$$

which leads to a single definition on the whole domain:

$$\begin{aligned}(\mathbf{KG}h)(\mathbf{y}\to\mathbf{x}) = &\int_{\mathcal{V}^\circ} \bar{f}\,(\mathbf{z}\to\mathbf{y}\to\mathbf{x})\,\bar{G}(\mathbf{y}\leftrightarrow\mathbf{z})\,h(\mathbf{z}\to\mathbf{y})\,\mathrm{d}V(\mathbf{z}) \\ &+\int_{\mathcal{M}} \bar{f}\,(\mathbf{z}\to\mathbf{y}\to\mathbf{x})\,\bar{G}(\mathbf{y}\leftrightarrow\mathbf{z})\,h(\mathbf{z}\to\mathbf{y})\,\mathrm{d}A(\mathbf{z}).\end{aligned}$$

Inserting increasing powers of **KG** into Equation (23) leads to a nested integral, which has the following compact representation on path space:

$$I_j = \int_{\mathcal{P}} f_j(\bar{x})\,\mathrm{d}\mu(\bar{x}) \tag{25}$$

where

$$\begin{aligned}f_j(\bar{\mathbf{x}}) = f_j(\mathbf{x}_1\cdots\mathbf{x}_n) = L_e(\mathbf{x}_1\to\mathbf{x}_2)\left[\prod_{k=2}^{n-1}\bar{f}(\mathbf{x}_{k-1}\to\mathbf{x}_k\to\mathbf{x}_{k+1})\,\bar{G}(\mathbf{x}_{k-1}\leftrightarrow\mathbf{x}_k)\right] \\ \cdot\,\bar{G}(\mathbf{x}_{n-1}\leftrightarrow\mathbf{x}_n)\,W_e^{(j)}(\mathbf{x}_{n-1}\to\mathbf{x}_n).\end{aligned} \tag{26}$$

The seeming direction reversal in the importance emission function is due to the convention that "$\to$" indicates the flow of light.

7 Manifold exploration for volumes

There are no specular reflections *per se* in the volume, but from a purely mathematical standpoint the phase function of a strongly forward-scattering volume is not unlike reflection from a rough mirror. The fast-varying part of the mirror BRDF is a function of the half-vector, and hence our method preserves it during manifold walks. In the medium case, we are interested in being able to handle highly peaked phase functions that are a function of the scattering angle. For this purpose we treat the scattering angle as analogous to the half vector, introducing the specular manifold constraint

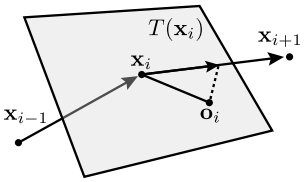


Figure 12: Medium constraint

$$\mathbf{c}(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1}) = T(\overrightarrow{\mathbf{x}_{i-1}\mathbf{x}_i})^T\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}} \quad (=\mathbf{o}_i) \tag{27}$$

where $T(\mathbf{v})$ is a basis for the plane orthogonal to the direction $\mathbf{v}$ (Figure 12). This still leaves one DOF per vertex, which we remove by preserving the distance to the scattering event (i.e. $\|\mathbf{x}_{i-1}-\mathbf{x}_i\| = \text{const.}$). While computing the entries of the constraint Jacobian ∇C, we use these two constraints in place of the previous definition (Section 3.3) whenever a vertex describes a medium interaction.

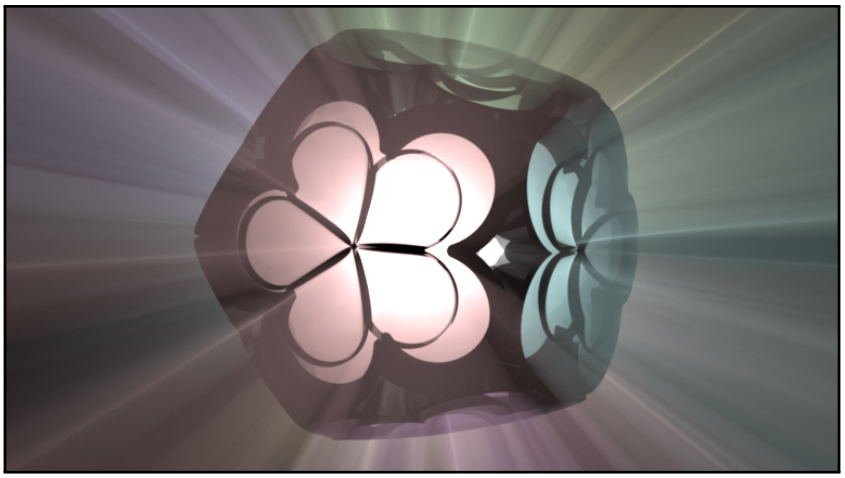


Figure 13: It is possible to extend the path space specular integration framework to volumes, for instance to render caustics from refractive objects, such as this dodecahedron-shaped luminaire with tinted glass inlays.

7.1 Medium manifold perturbation

From an algorithmic perspective, manifold exploration for volumes is almost identical to the surface case. Our implementation handles both cases jointly, which permits constructing specular chains that contain both surface and medium interactions.

Apart from the new type of constraint (27), the computation of offset manifold (11) tangent vectors is unchanged. In the ray tracing step 4 of WalkManifold when encountering a vertex $\mathbf{x}_{i-1}$ that is followed by a medium interaction vertex $\mathbf{x}_i$, we set $\mathbf{x}'_i = \mathbf{x}_{i-1} + \|\mathbf{x}_{i-1} - \mathbf{x}_i\| \, \mathbf{d}$, where $\mathbf{d}$ is the outgoing direction at $\mathbf{x}_{i-1}$ (this enforces the length constraint mentioned earlier). Afterwards, the manifold offset $\mathbf{o}_i$ is transformed into an outgoing direction in the new frame at $\mathbf{x}'_i$ (Figure 12).

As in the glossy surface case, we require a criterion that clarifies when treating a medium vertex $\mathbf{x}_i$ as non-specular is in order, and when it is better handled by the manifold. Again, this decision is made probabilistically, based on a modified specular probability function $\psi(\mathbf{x}_i)$ described in the appendix.

When computing the generalized geometric term through a specular chain with medium endpoints, we consider them to be parameterized on a surface perpendicular to the direction facing the chain.

8 Results

We have implemented the proposed technique and prior work as extension modules to the Mitsuba renderer [Jakob 2010]. All techniques operate on top of a newly added *bidirectional abstraction layer* that exposes cameras, light sources, scattering models, and participating media as generalized path vertex and edge objects with a common basic interface. This greatly simplified the implementation effort, as bidirectional rendering algorithms can usually be stated much more succinctly in terms of operations on vertices that are oblivious to whether they contain e.g. a camera model or a medium scattering event.

Scene	Seed generator		Perturb.	Mutations		Manifold walks			Manif. size	
	samples/px.	chains/px.	λ	total	accepted	total	conv.	avg. iter.	avg.	max.
Torus (MEPT)	32	2	100	1178 M	78.3%	1002 M	96.7%	2.3	3.4	7
Chandelier (MEPT)	64	3	160	1216 M	73.6%	975 M	97.6%	2.4	4.6	13
Chandelier (MEMLT)	—	—	160	3626 M	34.1%	376 M	98.4%	2.1	4.5	13
Table (MEPT)	32	1	300	1074 M	77.5%	868 M	95.5%	2.8	4.4	14
Table (MEMLT)	—	—	300	1921 M	35.9%	772.4 M	94.3%	3.2	4.4	14
GlassEgg (MEPT)	128	2	90	1533 M	72.4%	1246 M	92.2%	3.4	4.1	14
GlassEgg (MEMLT)	—	—	90	2774 M	40.3%	1101 M	92.5%	3.3	4.1	14

Table 1: Listing of seed generator and perturbation parameters, as well as performance statistics.

We compare the following algorithms:

- Primary sample space MLT by Kelemen et al., implemented on top of bidirectional path tracing (PSSMLT).
- Path space MLT by Veach and Guibas (MLT).
- An extended form of energy redistribution path tracing by Cline et al. (ERPT), which is seeded by bidirectional rather than unidirectional path tracing. The ERPT implementation shares the caustic, lens, and multi-chain perturbation with the previous algorithm. Since they introduce bias, we did not use the post-processing filters proposed in the original paper.
- Manifold exploration path tracing (MEPT), which is structured similarly to ERPT. We modified the original algorithm by replacing its highly specialized caustic, lens, and multi-chain perturbations with the manifold perturbation. Due to its general design, the new perturbation subsumes and extends the capabilities of the original set.
- Manifold Metropolis Light Transport (MEMLT), which corresponds to MLT with our perturbation (i.e. the bidirectional mutation and manifold perturbation, but none of the original perturbations from MLT).

Due to the aforementioned abstraction layer, all techniques transparently support participating media even if this was originally not part of their description. We found that MEPT generally performs better that MEMLT due to certain limitations of the bidirectional mutation that are discussed later.

The rendering of result images was conducted on Amazon EC2 `cc1.4xlarge` instances, which are eight-core Intel Xeon X5570 machines. A single machine was used per image. To exploit the local parallelism, our implementation runs a separate Markov chain on each core, and the resulting buffers are averaged together when exposing the image.

We have rendered three views of a challenging interior scene containing approximately 2 million triangles with shading normals and a mixture of glossy, diffuse, and specular surfaces and some scattering volumes. One hour of processing time was allocated to each rendering technique, and a comparison of the resulting images is shown in Figures 14, 15, and 16. The converged images in Figure 1 were rendered using 48 hours. The one hour renderings are intentionally unconverged to permit a visual analysis of the convergence behavior. Table 1 lists parameters and statistics collected during these renderings. The path generator columns refer to the seeding scheme used by ERPT and MEPT, which samples and subsequently resamples a number of paths

per pixel before launching Markov Chains. The statistics include the total number of mutations and acceptance ratio, as well as the the convergence behavior of the manifold walks and vertex count of encountered manifolds.

Chandelier: In this set of results, the poor performance of MLT is most apparent and is caused by the ineffectiveness of the bidirectional mutation in finding long specular paths. Because it must decide up front on the configuration of a path before generating it, most of the time the mutation fails, resulting in acceptance rates under 1%. Consequently, too few jumps between disjoint connected components of path space occur, causing parts of the image have an incorrect relative brightness. This weakness is inherited by MEMLT, which also builds upon the bidirectional mutation. It is more successful at exploring some of the diffuse-specular-diffuse paths through the bulbs but overall does not work well on this scene. Seeding the same perturbations using bidirectional path tracing, which does not suffer from this disadvantage, performs much better, as can be seen in the ERPT and MEPT renderings.

Table: This scene is lit by the chandelier, with its glass-enclosed sources, so all illumination is by specular paths. By reasoning about the geometry of the specular and offset specular manifolds for the paths it encounters, our perturbation rule is more successful at rendering paths—such as illumination that refracts from the bulbs into the butter dish, then to the camera (6 specular vertices)—that the other methods struggle with. The MLT rendering looks too dark, because it did not find enough of these paths and mainly captures diffuse illumination from the walls. The noise in the ERPT result reveals that the underlying bidirectional path tracer encountered some of those paths but the Veach–Guibas perturbations are not able to explore path space around them effectively. The primary sample space MLT variant also has difficulties rendering this scene, because it has no knowledge about the underlying path geometry. MEMLT produces a clean result, but the relative brightness of different parts of the image is incorrect due to bidirectional mutation's difficulty in performing sufficiently many jumps between them.

GlassEgg: In this scene, our technique's ability to create a specular chain containing both medium and surface interactions leads to fast convergence when rendering the forward-scattering medium inside the glass egg. MLT and ERPT perform poorly here, since they do not have suitable perturbations for exploring this space. Because the MLT perturbations treat glossy and diffuse materials identically, they have difficulty rendering the near-specular tabletop, producing streak-like artifacts in the output rendering.

9 Conclusions

We have presented a new type of Markov Chain Monte Carlo rendering method that models the space of valid specular and near-specular light paths using high-dimensional manifolds. The differential geometry of these manifolds provides a powerful tool to efficiently explore these paths, which can be very hard for previous methods to find. Our technique applies in the frameworks of MLT or ERPT, producing rendering algorithms with support for specular paths fundamentally built in at the core. In equal-time comparisons on very challenging scenes, the new Manifold Exploration Path Tracing algorithm compares favorably to previous work in Monte Carlo and MCMC rendering.

This new algorithm does still share certain limitations with its predecessors. Most important, it needs well distributed seed paths, because it can only explore connected components of the

Figure 14: CHANDELIER: This view contains a brass chandlier with 24 light bulbs, each surrounded by a glass enclosure. The chandlier uses a realistic metal material based on microfacet theory and is attached to the ceiling using specular metal cylinders. This scene is challenging, as certain important light paths are found with low probability, particularly those involving interreflection between the bulbs and the body of the chandelier. In this and the following comparisons, one hour of processing time was allocated to each rendering technique.

Figure 14: Chandelier (continued)

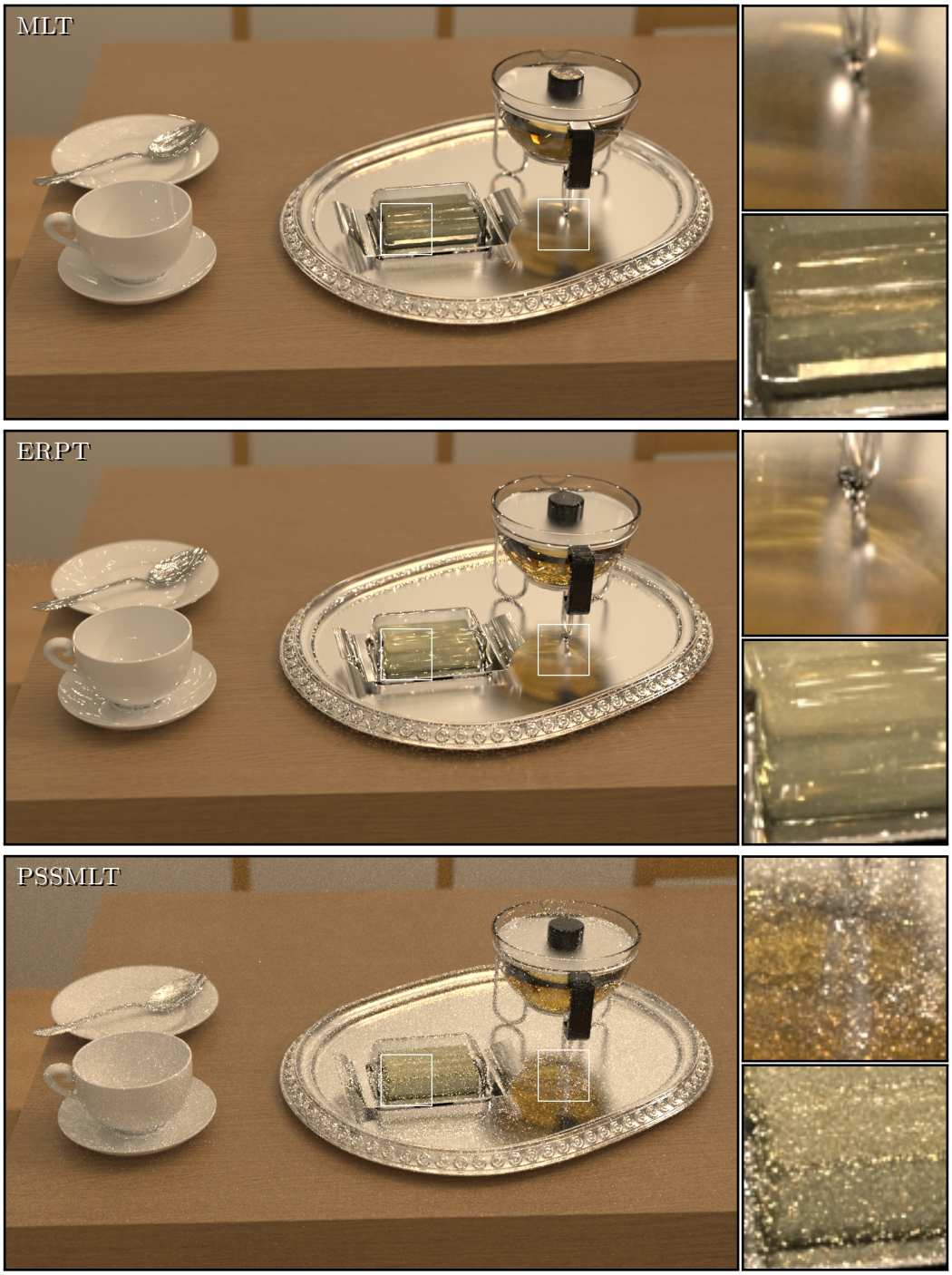


Figure 15: TABLE: This view of our room scene shows chinaware (using a BRDF with both diffuse and specular components), a teapot containing an absorbing medium, and a butter dish on a glossy silver tray. Illumination comes from the chandelier in Figure 14.

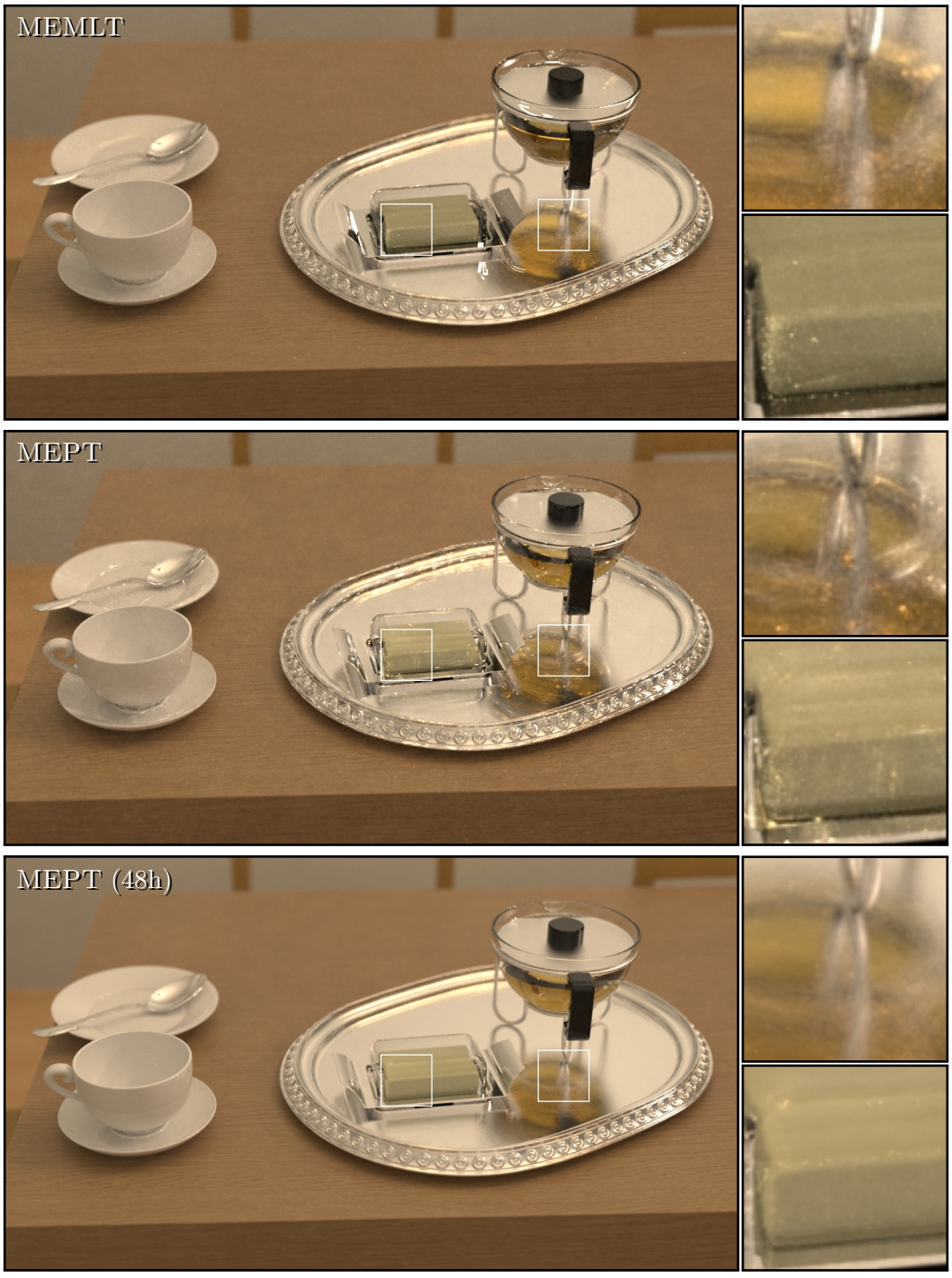


Figure 15: TABLE (continued)

Figure 16: GlassEgg: This view of a different part of the room, now lit through windows using a spherical environment map surrounding the scene, contains a forward-scattering participating medium ($g = 0.8$) inside the glass egg.

manifold for which seed paths are provided. Bidirectional Path Tracing is reasonably effective but still has trouble finding many components, and this problem fundamentally becomes more and more difficult as the number of path types increases. Ultimately, as the number of different path types exceeds the number of samples that can be generated, local exploration of path space becomes ineffective; future algorithms could be designed to attempt exploration only in sufficiently large path space components.

Unlike many methods for caustics and other specular phenomena, we have shown how to generalize Manifold Exploration almost trivially to handle glossy surfaces and volumes. Similar refinements can let the same method handle perfectly anisotropic reflections, strongly oriented volume scattering media, and other kinds of problems with exactly or approximately constrained paths.

While MCMC rendering is a natural match for our methods of dealing with specular paths, our predictor-corrector iteration can be used in other kinds of algorithms as well, including deterministic ones to map out specular paths, for instance in design of luminaires or optical systems.

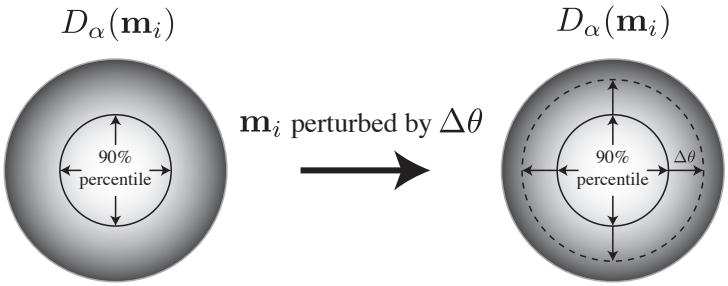
Figure 16: GLASSEGG (continued)

10 Acknowledgements

This research was conducted in conjunction with the Intel Science and Technology Center for Visual Computing. Additional funding was provided by the National Science Foundation under grant IIS-1011919. The authors are indebted to Olesya Isaenko, who meticulously crafted the example scenes used in the evaluation.

Appendix A1

Specular probability function for surfaces: We found the following following heuristic based on microfacet theory to work well: when the microsurface normals at $\mathbf{x}_i$ follow a distribution $D_\alpha(\mathbf{m}_i)$ with roughness parameter α, the BSDF at $\mathbf{x}_i$ will take on small values when $\mathbf{m}_i$ moves into a region where $D_\alpha(\mathbf{m}_i)$ has low density. We thus set $\psi(\mathbf{x}_i)$ by computing the expected probability that treating vertex $\mathbf{x}_i$ as non-specular during a manifold perturbation would move its microsurface normal $\mathbf{m}_i$ from a region of high density to one of low density, and we choose the 90th-percentile to classify the support of D_α into such regions.



To obtain the specular probability, our implementations must know the expected angular change $\Delta\theta$ of microsurface normals during a perturbation, which is found by briefly running the Markov chain before rendering starts. During rendering, $\psi(\mathbf{x}_i)$ is computed as the area ratio of the two highlighted regions on the sphere:

$$\psi(\mathbf{x}_i) = \frac{1 - \cos\theta_q(\alpha(\mathbf{x}_i))}{1 - \cos\left(\theta_q(\alpha(\mathbf{x}_i)) + \Delta\theta\right)}, \tag{28}$$

where θ_q is the aforementioned percentile (with q set to 0.9). For the Beckmann distribution, this is given by

$$\theta_q(\alpha) \coloneqq \tan^{-1}(-\alpha^2 \log(1-q)).$$

Implementation-wise, this heuristic requires material models to be able to compute their Beckmann distribution-equivalent roughness or provide a custom quantile function.

Specular probability function for participating media: In the medium case, we use the same probability (28), but now with a percentile that is suitable for volumetric scattering. We use

$$\theta_q(g) = \cos^{-1}\frac{(1+|g|)^2 - 2(1+|g|)(1+g^2)q + 2|g|(1+g^2)q^2}{(1+|g|-2|g|q)^2}$$

where g is the *mean cosine* of the phase function, q is set to 0.5, and θ_q was derived from the Henyey-Greenstein phase function. In $\Psi(\mathbf{x}_i)$ (Equation 28) we must also replace $\Delta\theta$ with the average change in scattering angle at medium vertices, again determined in a brief phase before rendering.

Appendix A2

Let us assume the existence of a `Vertex` data structure with the following contents:

```
struct Vertex {
    Point p;              // Vertex position
    Vector dpdu, dpdv;    // Tangent vectors
    Normal n;             // Normal vector
    Vector dndu, dndv;    // Normal derivatives
    float eta;            // Relative index of refraction
    Matrix2x2 A, B, C;    // Matrix blocks of ∇C in the row
                          // associated with the current vertex
};
```

Then the following C++ code computes the derivatives of ∇C associated with a single surface interaction vertex. It assumes that the function is called with a pointer into a list of vertices.

```
void computeDerivatives(Vertex *v) {
    /* Compute relevant directions and a few useful projections */
    Vector wi = v[-1].p - v[0].p;
    Vector wo = v[ 1].p - v[0].p;
    float ili = 1/wi.length();
    float ilo = 1/wo.length();
    wi *= ili; wo *= ilo;

    Vector H = wi + v[0].eta * wo;
    float ilh = 1/H.length();
    H *= ilh;

    float dot_H_n    = dot(v[0].n, H),
          dot_H_dndu = dot(v[0].dndu, H),
          dot_H_dndv = dot(v[0].dndv, H),
          dot_u_n    = dot(v[0].dpdu, v[0].n),
          dot_v_n    = dot(v[0].dpdv, v[0].n);

    /* Local shading tangent frame */
    Vector s = v[0].dpdu - dot_u_n * v[0].n;
    Vector t = v[0].dpdv - dot_v_n * v[0].n;

    ilo *= v[0].eta * ilh; ili *= ilh;

    /* Derivatives of C with respect to x_{i-1} */
    Vector
        dH_du = (v[-1].dpdu - wi * dot(wi, v[-1].dpdu)) * ili,
        dH_dv = (v[-1].dpdv - wi * dot(wi, v[-1].dpdv)) * ili;

    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].A = Matrix2x2(
        dot(dH_du, s), dot(dH_dv, s),
        dot(dH_du, t), dot(dH_dv, t));

    /* Derivatives of C with respect to x_i */
    dH_du = -v[0].dpdu * (ili + ilo) + wi * (dot(wi, v[0].dpdu) * ili)
                                     + wo * (dot(wo, v[0].dpdu) * ilo);
    dH_dv = -v[0].dpdv * (ili + ilo) + wi * (dot(wi, v[0].dpdv) * ili)
                                     + wo * (dot(wo, v[0].dpdv) * ilo);
    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].B = Matrix2x2(
        dot(dH_du, s) - dot(v[0].dpdu, v[0].dndu) * dot_H_n - dot_u_n * dot_H_dndu,
        dot(dH_dv, s) - dot(v[0].dpdu, v[0].dndv) * dot_H_n - dot_u_n * dot_H_dndv,
        dot(dH_du, t) - dot(v[0].dpdv, v[0].dndu) * dot_H_n - dot_v_n * dot_H_dndu,
```

```
        dot(dH_dv, t) - dot(v[0].dpdv, v[0].dndv) * dot_H_n - dot_v_n * dot_H_dndv);

    /* Derivatives of C with respect to x_{i+1} */
    dH_du = (v[1].dpdu - wo * dot(wo, v[1].dpdu)) * ilo;
    dH_dv = (v[1].dpdv - wo * dot(wo, v[1].dpdv)) * ilo;

    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].C = Matrix2x2(
        dot(dH_du, s), dot(dH_dv, s),
        dot(dH_du, t), dot(dH_dv, t));
}
```

References

Arvo, J. R. 1995. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University.

Chen, M., and Arvo, J. 2000. Perturbation methods for interactive specular reflections. *IEEE Trans. Vis. and Comp. Graph. 6*, 3 (July/Sept.), 253–264.

Chen, M., and Arvo, J. 2000. Theory and application of specular path perturbation. *ACM Trans. Graph. 19*, 4 (Oct.), 246–278.

Chen, J., Wang, B., and Yong, J.-H. 2011. Improved stochastic progressive photon mapping with metropolis sampling. *Computer Graphics Forum 30*, 4, 1205–1213.

Cline, D., Talbot, J., and Egbert, P. 2005. Energy redistribution path tracing. *ACM Trans. Graph. 24*, 3 (Aug.), 1186–1195.

Cook, R. L., Porter, T., and Carpenter, L. 1984. Distributed ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, 137–145.

Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. 1984. Modeling the interaction of light between diffuse surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, 213–222.

Hachisuka, T., and Jensen, H. W. 2009. Stochastic progressive photon mapping. *ACM Trans. Graph. 28*, 5 (Dec.).

Hachisuka, T., and Jensen, H. W. 2011. Robust adaptive photon tracing using photon path visibility. *ACM Trans. Graph. 30*, 5 (Oct.), 114:1–114:11.

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

Heckbert, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, 145–154.

IGEHY, H. 1999. Tracing ray differentials. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, 179–186.

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

JAROSZ, W., ZWICKER, M., AND JENSEN, H. W. 2008. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum 27*, 2 (Apr.), 557–566.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Computer Graphics (Proceedings of SIGGRAPH 98)*, 311–320.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Eurographics Rendering Workshop 1996*, 21–30.

KAJIYA, J. T. 1986. The rendering equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, 143–150.

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

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

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In *Proceedings of Compugraphics 93*.

LAI, Y.-C., FAN, S. H., CHENNEY, S., AND DYER, C. 2007. Photorealistic image rendering with population monte carlo energy redistribution. In *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, 287–296.

LIU, J. S. 2001. *Monte Carlo strategies in scientific computing*. Springer.

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.

MITCHELL, D. P., AND HANRAHAN, P. 1992. Illumination from curved reflectors. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 283–291.

PAULY, M., KOLLIG, T., AND KELLER, A. 2000. Metropolis light transport for participating media. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, 11–22.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

PREMOZE, S., ASHIKHMIN, M., AND SHIRLEY, P. 2003. Path integration for light transport in volumes. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*.

SEGOVIA, B., IEHL, J., AND PÉROCHE, B. 2007. Metropolis instant radiosity. *Computer Graphics Forum 26*, 3 (Sept.), 425–434.

Shirley, P. S., Wade, B., Hubbard, P., Zareski, D., Walter, B., and Greenberg, D. P. 1995. Global illumination via density estimation. In *Eurographics Rendering Workshop 1995*, 219–231.

Sillion, F. X., and Puech, C. 1989. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, 335–344.

Spivak, M. 1965. *Calculus on Manifolds*. Addison-Wesley.

Veach, E., and Guibas, L. 1994. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*.

Veach, E., and Guibas, L. J. 1997. Metropolis light transport. In *Computer Graphics (Proceedings of SIGGRAPH 97)*, 65–76.

Veach, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

Walter, B., Hubbard, P. M., Shirley, P. S., and Greenberg, D. F. 1997. Global illumination using local linear density estimation. *ACM Trans. Graph. 16*, 3 (July), 217–259.

Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. 2007. Microfacet models for refraction through rough surfaces. In *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, 195–206.

Walter, B., Zhao, S., Holzschuch, N., and Bala, K. 2009. Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph. 28*, 3 (July), 92:1–92:8.

Whitted, T. 1980. An improved illumination model for shaded display. *Communications of the ACM 23*, 6 (June), 343–349.