Path Space Markov Chain Monte Carlo Methods in Computer Graphics

Wenzel Jakob

Abstract The objective of a rendering algorithm is to compute a photograph of a simulated reality, which entails finding all the paths along which light can flow from a set of light sources to the camera. The purpose of this article is to present a high-level overview of the underlying physics and analyze how this leads to a high-dimensional integration problem that is typically handled using Monte Carlo methods. Following this, we survey recent work on path space Markov Chain Monte Carlo (MCMC) methods that compute the resulting integrals using proposal distributions defined on sets of light paths.

1 Introduction

The central goal of light transport algorithms in computer graphics is the generation of *renderings*, two-dimensional images that depict a simulated environment as if photographed by a virtual camera. Driven by the increasing demand for photorealism, computer graphics is currently undergoing a substantial transition to physics-based rendering techniques that compute such images while accurately accounting for the interaction of light and matter.

These methods require a detailed model of the scene including the shape and optical properties of all objects including light sources; the final rendering is then generated by a simulation of the relevant physical laws, specifically *transport* and *scattering*, i.e., the propagation of light and its interaction with the materials that comprise the objects. In this article, we present a high-level overview of the underlying physics and analyze how this leads to a high-dimensional integration problem that is typically handled using Monte Carlo methods.

Section 2 begins with a discussion of the geometric optics framework used in computer graphics. After defining the necessary notation and physical units, we

Interactive Geometry Lab, ETH Zürich, e-mail: wjakob@inf.ethz.ch

state the energy balance equation that characterizes the interaction of light and matter. Section 3 presents a simple recursive Monte Carlo estimator that solves this equation, though computation time can be prohibitive if accurate solutions are desired. Section 4 introduces path space integration, which offers a clearer view of the underlying light transport problem. This leads to a large class of different estimators that can be combined to improve convergence. Section 5 introduces MCMC methods in rendering. Section 6 covers an MCMC method that explores a lower-dimensional manifold of light paths, and Section 7 discusses extensions to cases involving inter-reflection between glossy objects. Section 8 concludes with a discussion of limitations and unsolved problems.

This article is by no means a comprehensive treatment of rendering; the selection of topics is entirely due to the author's personal preference. It is intended that the discussion will be helpful to readers who are interested in obtaining an understanding of recent work on path-space methods and applications of MCMC methods in rendering.

2 Geometric Optics and Light Transport on Surfaces

Light transport simulations in computer graphics are generally conducted using a simplified variant of geometric optics. In this framework, light moves along a straight line until an interaction (i.e., a scattering event) occurs, which involves a change of direction and potentially some absorption. The wave-like nature of light is not simulated, which leads to a simpler computation and is an excellent approximation in general (the wavelength of visible light is minuscule compared to the sizes of everyday objects). Light is also assumed to be incoherent and unpolarized, and although it moves at a finite speed, this motion is not modeled explicitly. More complex theories without these assumptions are available but ultimately not needed since the phenomena described by them are in most cases too subtle to be observed by humans. For the sake of simplicity, we only discuss monochromatic rendering in this article; the generalization to the full color spectrum poses no fundamental difficulties.

In the following sections, we review relevant background material, starting with the standard light transport model used in computer graphics and leading up to the path space framework proposed by Veach [28].

In geometric optics, light is usually quantified using *radiance*, which has units of $\mathrm{W} \cdot \mathrm{sr}^{-1} \cdot \mathrm{m}^{-2}$. Given a point $\mathbf{x} \in \mathbb{R}^3$ and a direction $\omega \in S^2$, the radiance $L(\mathbf{x}, \omega)$ is a density function that describes how much illumination flows through the point, in this direction. Radiance can be measured by registering the amount of energy arriving on a small surface patch $\mathrm{d}A$ at $\mathbf{x}$ that is perpendicular to ω and sensitive to a small cone of directions $\mathrm{d}\omega$ around ω, and then letting the surface and solid angle tend to zero. For a thorough review of radiance and many related radiometric quantities, we refer the reader to Preisendorfer [25].

An important property of radiance is that it remains invariant along rays when there are no obstructions (e.g., in vacuum),

$$L(\mathbf{x},\omega) = L(\mathbf{x}+t\omega,\,\omega), \quad t \in [0, t_{\text{obstr}}).$$

Due to this property, a complete model of a virtual environment can be obtained simply by specifying how L behaves in places where an obstruction interacts with the illumination, i.e., at the boundaries of objects or inside turbid substances like fog or milk. In this article, we only focus on the boundary case for simplicity. For a more detailed discussion including volumetric scattering, we refer to [10].

We assume that the scene to be rendered is constructed from a set of surfaces that all lie inside a bounded domain $\Omega \subseteq \mathbb{R}^3$. The union of these surfaces is denoted $\mathscr{M} \subset \Omega$ and assumed to be a differentiable manifold, i.e. is parameterized by a set of charts with differentiable transition maps.

Furthermore, let $N : \mathscr{M} \to S^2$ denote the Gauss map, which maps surface positions to normal directions on the unit sphere.

Because boundaries of objects introduce discontinuities in the radiance function L, we must take one-sided limits to distinguish between the exterior radiance function $L^+(\mathbf{x},\omega)$ and the interior radiance function $L^-(\mathbf{x},\omega)$ at surface locations $\mathbf{x} \in \mathscr{M}$ as determined by the normal $N(\mathbf{x})$ (Figure 1). Based on these limits, intuitive incident and outgoing radiance functions can then be defined as

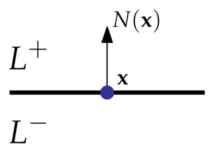


Fig. 1: *Limits of the radiance function L from above and below*

$$L_i(\mathbf{x},\omega) := \begin{cases} L^+(\mathbf{x},-\omega), & \omega\cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x},-\omega), & \omega\cdot N(\mathbf{x}) < 0 \end{cases} \qquad \text{and}$$

$$L_o(\mathbf{x},\omega) := \begin{cases} L^+(\mathbf{x},\omega), & \omega\cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x},\omega), & \omega\cdot N(\mathbf{x}) < 0 \end{cases}.$$

With the help of these definitions, we can introduce the surface energy balance equation that describes the relation between the incident and outgoing radiance based on the material properties at $\mathbf{x}$:

$$L_o(\mathbf{x},\omega) = \int_{S^2} L_i(\mathbf{x},\omega')\, f(\mathbf{x},\omega'\to\omega)\, \left|\omega'\cdot N(\mathbf{x})\right| \,\mathrm{d}\omega' + L_e(\mathbf{x},\omega), \quad \mathbf{x}\in\mathscr{M}. \quad (1)$$

The integration domain S^2 is the unit sphere and f is the *bidirectional scattering distribution function* (BSDF) of the surface, which characterizes the surface's response to illumination from different directions. Given illumination reaching a point $\mathbf{x}$ from a direction ω', the BSDF expresses how much of this illumination is scattered into the direction ω. For a detailed definition of the concept of a BSDF as well as other types of scattering functions, we refer the reader to Nicodemus [22]. The function $L_e(\mathbf{x},\omega)$ is the source term which specifies how much light is emitted from position $\mathbf{x}$ into direction ω; it is zero when the position $\mathbf{x}$ is not located on a light source.

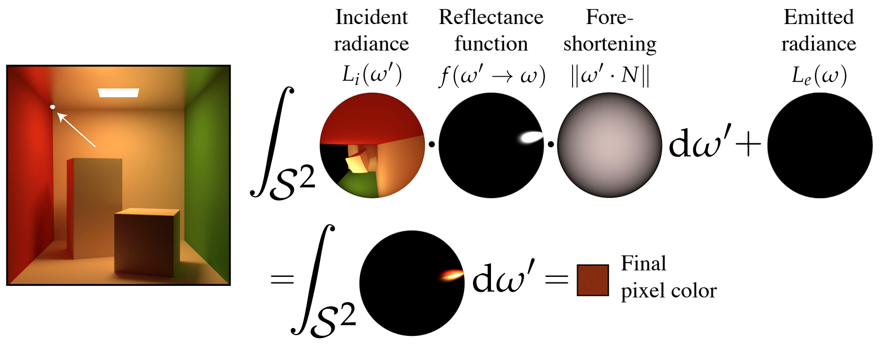

Incident radiance $L_i(\omega')$
Reflectance function $f(\omega' \to \omega)$
Fore-shortening $\|\omega' \cdot N\|$
Emitted radiance $L_e(\omega)$
$\int_{\mathcal{S}^2}$ · · $\mathrm{d}\omega' +$
$= \int_{\mathcal{S}^2}$ $\mathrm{d}\omega' =$
Final pixel color

Fig. 2: Illustration of the energy balance equation (1) on surfaces. Here, it is used to compute the pixel color of the surface location highlighted in white (only the top hemisphere is shown in the figure).

Figure 2 visualizes the different terms in Equation (1) over the top hemisphere. The example shows a computation of the radiance traveling from the surface location marked with a white dot towards the camera. The first term is an integral over the incident radiance as seen from the surface location. The integral also contains the BSDF and a cosine foreshortening term which models the effect that a beam of light arriving at a grazing angle spreads out over a larger region on the receiving surface and thus deposits less energy per unit area. The "ceiling" of the scene is made of rough metal; its reflectance function effectively singles out a small portion of the incident illumination, which leads to a fairly concentrated reflection compared to the other visible surfaces. The emission term is zero, since the highlighted surface position is not located on a light source.

Considerable research has been conducted on characterizing the reflectance properties of different materials, and these works have proposed a wide range of BSDF functions f that reproduce their appearance in renderings. Figure 3 shows several commonly used BSDF models, along with the resulting material appearance. The illustrations left of the renderings show polar plots of the BSDF $f(\omega' \to \omega)$ where the surface receives illumination from a fixed incident direction ω' highlighted in red. The primary set of reflected directions is shown in blue, and the transmitted directions (if any) are shown in green.

Specular materials shown in the top row are characterized by having a "degenerate" BSDF f that is described by a Dirac delta distribution. For instance, a mirror reflects light arriving from ω into only a single direction $\omega' = 2N(\mathbf{x})(\omega \cdot N(\mathbf{x})) - \omega$. In comparison, rough materials usually have a smooth function f. BSDFs based on *microfacet theory* [27, 4, 32] are a popular choice in particular—they model the interaction of light with random surfaces composed of tiny microscopic facets that are oriented according to a statistical distribution. Integration over this distribution then leads to simple analytic expressions that describe the expected reflection and refraction properties at a macroscopic scale. In this article, we assume that the

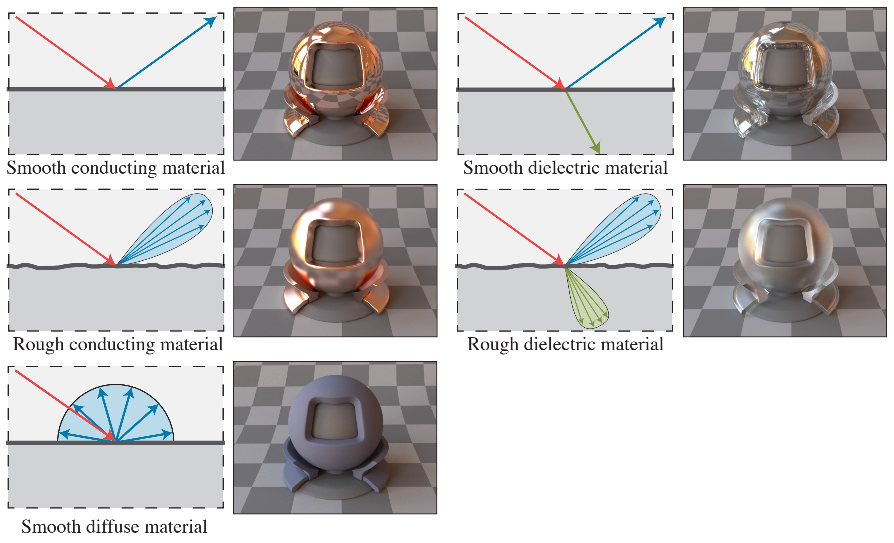

Smooth conducting material
Smooth dielectric material
Rough conducting material
Rough dielectric material
Smooth diffuse material

Fig. 3: An overview of common material types. The left side of each example shows a 2D illustration of the underlying scattering process for light arriving from the direction highlighted in red. The right side shows a corresponding rendering of a material test object.

BSDFs are provided as part of the input scene description and will not discuss their definitions in detail.

3 Path Tracing

We first discuss how Equation (1) can be solved using Monte Carlo integration, which leads to a simple method known as *Path Tracing* [12]. For this, it will be convenient to establish some further notation: we define the distance to the next surface encountered by the ray $(\mathbf{x},\omega)\in\mathbb{R}^3\times S^2$ as

$$d_{\mathscr{M}}(\mathbf{x},\omega) := \inf\{d>0 \mid \mathbf{x}+d\omega\in\mathscr{M}\}$$

where $\inf\emptyset=\infty$. Based on this distance, we can define a *ray-casting function* r:

$$r(\mathbf{x},\omega) := \mathbf{x}+d_{\mathscr{M}}(\mathbf{x},\omega)\,\omega. \tag{2}$$

Due to the preservation of radiance along unoccluded rays, the ray-casting function can be used to relate the quantities L_i and L_o:

$$L_i(\mathbf{x},\omega) = L_o(r(\mathbf{x},\omega),-\omega).$$

Algorithm 1 Pseudocode of a simple Path Tracer

```
function $\widehat{L}_o(\mathbf{x},\omega)$
    Return zero with probability $\alpha \in (0,1)$.
    Sample a direction $\omega'$ proportional to $f(\mathbf{x},\omega' \to \omega)$,
    let the factor of proportionality be denoted as $f_{\text{prop}}$.
    Set $\mathbf{x}' = r(\mathbf{x},\omega')$
    Return $\frac{1}{1-\alpha}\big(L_e(\mathbf{x},\omega) + f_{\text{prop}}\widehat{L}_o(\mathbf{x}',-\omega')\big)$.
```

In other words, to find the incident radiance along a ray $(\mathbf{x},\omega)$, we must only determine the nearest surface visible in this direction and evaluate its outgoing radiance into the opposite direction. Using this relation, we can eliminate L_i from the energy balance equation (1):

$$L_o(\mathbf{x},\omega) = \int_{S^2} L_o(r(\mathbf{x},\omega'),-\omega')\, f(\mathbf{x},\omega' \to \omega)\, \left|\omega' \cdot N(\mathbf{x})\right| \,\mathrm{d}\omega' + L_e(\mathbf{x},\omega) \qquad (3)$$

Although the answer is still not given explicitly, the equation is now in a form that is suitable for standard integral equation solution techniques. However, this is made difficult by the ill-behaved nature of the integrand, which is generally riddled with singularities and discontinuities caused by visibility changes in the ray-casting function r. Practical solution methods often rely on a Neumann series expansion of the underlying integral operator, in which case the resulting high number of dimensions rules out standard deterministic integration rules requiring an exponential number of function evaluations. Monte Carlo methods are resilient to these issues and hence see significant use in rendering.

To obtain an unbiased MC estimator based on Equation (3), we replace the integral with a single sample of the integrand at a random direction ω' and divide by its probability density $p(\omega')$, i.e.

$$\widehat{L}_o(\mathbf{x},\omega) = \frac{\widehat{L}_o(r(\mathbf{x},\omega'),-\omega')\, f(\mathbf{x},\omega' \to \omega)\, |\omega' \cdot N(\mathbf{x})|}{p(\omega')} + L_e(\mathbf{x},\omega) \qquad (4)$$

In this case, $E_p\widehat{L}_o = L_o$, and by averaging many estimates $\widehat{L}_o$, we obtain an approximation of the original integral. Typically, some form of importance sampling is employed, e.g. by choosing a sampling density function $p(\omega') \propto f(\mathbf{x},\omega' \to \omega)$. Algorithm 1 shows the pseudo-code of the resulting recursive method. Based on the underlying sequence of spherical sampling steps, path tracing can also be interpreted as a method that generates trajectories along which light is carried from the light source to the camera; we refer to these trajectories as a *light paths* and will revisit this concept in more detail later. In practice, the path tracing algorithm is combined with additional optimizations that lead to better convergence, but this is beyond the scope of this article.

Due to their simplicity and ability to produce photorealistic images, optimized path tracing methods have seen increased use in research and industrial applications.

The downside of these methods is that they converge very slowly given challenging input, sometimes requiring days or even weeks to compute a single image on state-of-the-art computers. Problems arise whenever complete light paths are found with too low a probability—a typical example is shown in Figure 5a.

4 The Path Space Formulation of Light Transport

In this section, we discuss the path space formulation of light transport, which provides a clearer view of the sampling operations performed by Algorithm 1. This framework can be used to develop other types of integration methods, including ones based on MCMC proposals that we discuss afterwards.

The main motivation for using path space is that it provides an *explicit* expression for the value of the radiance function as an integral over *light paths*, as opposed to the unwieldy recursive estimations on spherical domains in Algorithm 1. This allows for considerable freedom in developing and comparing sampling strategies. The path space framework was originally developed by Veach [28] and builds on a theoretical analysis of light transport operators by Arvo [1]. Here, we only present a high-level sketch.

Let us define an integral operator T

$$(Th)(\mathbf{x},\omega) := \int_{S^2} h(r(\mathbf{x},\omega'),-\omega')\, f(\mathbf{x},\omega'\to\omega)\, \left|\omega'\cdot N(\mathbf{x})\right|\, \mathrm{d}\omega', \tag{5}$$

and use it to rewrite Equation (3) as

$$L_o = TL_o + L_e.$$

An explicit solution for L_o can be found by inverting the operator so that

$$L_o = (1-T)^{-1}L_e.$$

Let $\|\cdot\|_L$ be a norm on the space of radiance functions

$$\|h\|_L := \int_{\mathscr{M}}\int_{S^2} h(\mathbf{x},\omega)\, |\omega\cdot N(\mathbf{x})|\,\mathrm{d}\omega\,\mathrm{d}A(\mathbf{x}),$$

which induces a corresponding operator norm $\|T\|_{\mathrm{op}} = \sup_{\|h\|_L\le 1}\|Th\|$. Veach proved that physically realizable scenes satisfy $\|T^l\|_{\mathrm{op}} < 1$ for some fixed $l\in\mathbb{N}$. Given this property, it is not only guaranteed that the inverse operator $(1-T)^{-1}$ exists, but it can also be computed using a Neumann series expansion:

$$(1-T)^{-1} = I + T + T^2 + \ldots,$$

which intuitively expresses the property that the outgoing radiance is equal to the emitted radiance plus radiance that has scattered one or more times (the sum converges since the energy of the multiply scattered illumination tends to zero).

$$L_o = L_e + TL_e + T^2 L_e + \cdots . \tag{6}$$

Rather than explicitly computing the radiance function L_o, the objective of rendering is usually to determine the response of a simulated camera to illumination that reaches its aperture. Suppose that the sensitivity of a pixel j in the camera is given by sensitivity profile function $W_e^{(j)} : \mathscr{M} \times S^2 \to \mathbb{R}$ defined on ray space. The intensity I_j of the pixel is given by

$$I_j = \int_{\mathscr{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega)\, L_o(\mathbf{r}(\mathbf{x}, \omega), -\omega)\; |\omega \cdot N(\mathbf{x})|\; \mathrm{d}\omega\, \mathrm{d}A(\mathbf{x}), \tag{7}$$

which integrates over its sensitivity function weighted by the outgoing radiance on surfaces that are observed by the camera. The spherical integral in the above expression involves an integrand that is evaluated at the closest surface position as seen from the ray $(\mathbf{x}, \omega)$. It is convenient to switch to a different domain involving only area integrals. We can transform the above integral into this form using the identity

$$\int_{S^2} q(\mathbf{r}(\mathbf{x}, \omega))\; |\omega \cdot N(\mathbf{x})|\; \mathrm{d}\omega = \int_{\mathscr{M}} q(\mathbf{y})\, G(\mathbf{x} \leftrightarrow \mathbf{y})\, \mathrm{d}A(\mathbf{y}), \tag{8}$$

where $\mathbf{x}, \mathbf{y} \in \mathscr{M}$, and $q : \mathscr{M} \to \mathbb{R}$ is any integrable function defined on surfaces, and G is the *geometric term* [24] defined as

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) := V(\mathbf{x} \leftrightarrow \mathbf{y}) \cdot \frac{\left|N(\mathbf{x}) \cdot \overrightarrow{\mathbf{xy}}\right| \left|N(\mathbf{y}) \cdot \overrightarrow{\mathbf{xy}}\right|}{\|\mathbf{x} - \mathbf{y}\|^2}. \tag{9}$$

The double arrows emphasize the symmetric nature of this function, $\overrightarrow{\mathbf{xy}}$ is the normalized direction from $\mathbf{x}$ to $\mathbf{y}$, and V is a visibility function defined as

$$V(\mathbf{x} \leftrightarrow \mathbf{y}) := \begin{cases} 1, & \text{if } \{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \mid 0 < \alpha < 1\} \cap \mathscr{M} = \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Applying the change of variables (8) to Equation (7) yields

$$I_j = \int_{\mathscr{M}} \int_{\mathscr{M}} W_e^{(j)}(\mathbf{x}, \overrightarrow{\mathbf{xy}})\, L_o(\mathbf{y}, \overrightarrow{\mathbf{yx}})\, G(\mathbf{x} \leftrightarrow \mathbf{y})\, \mathrm{d}A(\mathbf{x}, \mathbf{y}). \tag{11}$$

We can now substitute L_o given by Equation (6) into the above integral, which is a power series of the T operator (i.e. increasingly nested spherical integrals). Afterwards, we apply the change of variables once more to convert all nested spherical integrals into nested surface integrals. This is tedious but straightforward and leads to an explicit expression of I_j in terms of an infinite series of integrals over increasing Cartesian powers of $\mathscr{M}$.

These nested integrals over surfaces are due to the propagation of light along straight lines and changes of direction at surfaces, which leads to the concept of a *light path*. This can be thought of as the trajectory of a particle carrying an infinitesimal portion of the illumination. It is a piecewise linear curve $\bar{\mathbf{x}} = \mathbf{x}_1 \cdots \mathbf{x}_n$ with endpoints $\mathbf{x}_1$ and $\mathbf{x}_n$ and intermediate scattering vertices $\mathbf{x}_2, \ldots, \mathbf{x}_{n-1}$. The space of all possible light paths is a union consisting of paths with just the endpoints, paths that have one intermediate scattering event, and so on. More formally, we define path space as

$$\mathscr{P} := \bigcup_{n=2}^{\infty} \mathscr{P}_n, \text{ and}$$
$$\mathscr{P}_n := \{\mathbf{x}_1 \cdots \mathbf{x}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathscr{M}\}. \tag{12}$$

The nested integrals which arose from our manipulation of Equation (11) are simply integrals over light paths of different lengths, i.e.

$$I_j = \int_{\mathscr{P}_2} \varphi(\mathbf{x}_1\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_1, \mathbf{x}_2) \; + \int_{\mathscr{P}_3} \varphi(\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3)\, \mathrm{d}A(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) + \ldots. \tag{13}$$

Because some paths carry more illumination from the light source to the camera than others, the integrand $\varphi : \mathscr{P} \to \mathbb{R}$ is needed to quantify their "light-carrying capacity"; its definition varies based on the number of input arguments and is given by Equation (15). The total illumination I_j arriving at the camera is often written more compactly as an integral of φ over the entire path space, i.e.:

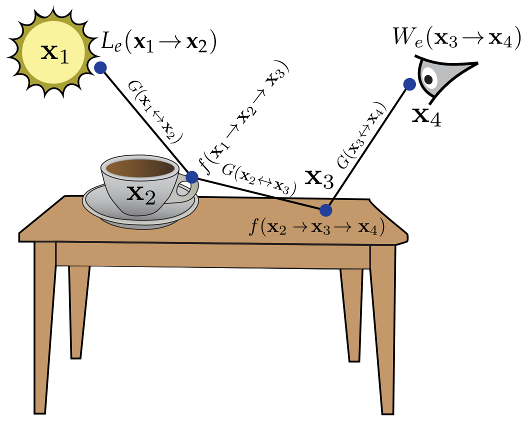
$$=: \int_{\mathscr{P}} \varphi(\bar{\mathbf{x}})\, \mathrm{d}A(\bar{\mathbf{x}}). \tag{14}$$

The definition of the weighting function φ consists of a product of terms—one for each vertex and edge of the path:

$$\varphi(\mathbf{x}_1 \cdots \mathbf{x}_n) = L_e(\mathbf{x}_1 \to \mathbf{x}_2) \left[\prod_{k=2}^{n-1} G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)\, f(\mathbf{x}_{k-1} \to \mathbf{x}_k \to \mathbf{x}_{k+1}) \right]$$
$$G(\mathbf{x}_{n-1} \leftrightarrow \mathbf{x}_n)\, W_e^{(j)}(\mathbf{x}_{n-1} \to \mathbf{x}_n). \tag{15}$$

The arrows in the above expression symbolize the symmetry of the geometric terms as well as the flow of light at vertices. $\mathbf{x}_i \to \mathbf{x}_{i+1}$ can also be read as a spatial argument $\mathbf{x}_i$ followed by a directional argument $\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}$. Figure 4 shows an example light path and the different weighting terms. We summarize their meaning once more:

- $L_e(\mathbf{x}_1 \to \mathbf{x}_2)$ is the emission profile of the light source. This term expresses the amount of radiance emitted from position $\mathbf{x}_1$ traveling towards $\mathbf{x}_2$. It is equal to zero when $\mathbf{x}_1$ is not located on a light source.
- $W_e^j(\mathbf{x}_{n-1} \to \mathbf{x}_n)$ is the sensitivity profile of pixel j of the camera; we can think of the pixel grid as an array of sensors, each with its own profile function.
- $G(\mathbf{x} \leftrightarrow \mathbf{y})$ is the geometric term (Equation 9), which specifies the differential amount of illumination carried along segments of the light path. Among other things, it accounts for visibility: when there is no unobstructed line of sight between $\mathbf{x}$ and $\mathbf{y}$, G evaluates to zero.


$\mathbf{x}_1$
$L_e(\mathbf{x}_1 \to \mathbf{x}_2)$
$W_e(\mathbf{x}_3 \to \mathbf{x}_4)$
$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$
$f(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3)$
$G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_4)$
$\mathbf{x}_4$
$G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3)$
$\mathbf{x}_3$
$\mathbf{x}_2$
$f(\mathbf{x}_2 \to \mathbf{x}_3 \to \mathbf{x}_4)$

$$\varphi(\bar{\mathbf{x}}) = L_e(\mathbf{x}_1 \to \mathbf{x}_2)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, f(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3)\, G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3)$$
$$f(\mathbf{x}_2 \to \mathbf{x}_3 \to \mathbf{x}_4)\, G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_4)\, W_e(\mathbf{x}_3 \to \mathbf{x}_4)$$

Fig. 4: Illustration of a simple light path with four vertices and its corresponding weighting function.

- $f(\mathbf{x}_{k-1} \to \mathbf{x}_k \to \mathbf{x}_{k+1})$ is the BSDF, which specifies how much of the light that travels from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ is then scattered towards position $\mathbf{x}_{k+1}$. This function essentially characterizes the material appearance of an object (e.g., whether it is made of wood, plastic, concrete, etc.).

Over the last 40 years, considerable research has investigated realistic expressions for the L_e, W_e, and f terms. In this article, we do not discuss their definition and prefer to think of them as black box functions that can be queried by the rendering algorithm. This is similar to how rendering software is implemented in practice: a scene description might reference a particular material (e.g., car paint) whose corresponding function f is provided by a library of material implementations. The algorithm accesses it through a high-level interface shared by all materials, but without specific knowledge about its internal characteristics.

4.1 *Regular Expressions to Select Sets of Light Paths*

Different materials can interact with light in fundamentally different ways, which has important implications on the design of rendering algorithms. It is helpful to distinguish between interactions using a 1-letter classification for each vertex type:

`S` (*ideal specular*): specular surfaces indicate boundaries between materials with different indices of refraction (e.g., air and water). Ideal specular boundaries have no roughness and cause and incident ray of light to be scattered into a *discrete* set of outgoing directions (Figure 3). Examples of specular materials include polished glass and metal surfaces and smooth coatings.

`G` (*glossy*): glossy surfaces also mark an index of refraction transition, but in this case the surface is affected by small-scale roughness. This causes the same ray to scatter into a continuous distribution of directions which concentrates around the same directions as the ideally smooth case.

`D` (*diffuse*): diffuse surfaces reflect light into a directional distribution that is either uniform or close to uniform; examples include clay and plaster.

We additionally assign the labels `L` and `E` to light source and camera ("eye") vertices, respectively, allowing for the classification of entire light paths using a sequence of symbols (e.g., "`LSDSE`"). Larger classes of paths can be described using Heckbert's *path regular expressions* [8], which add convenient regular expression rules such as the Kleene star "$\star$" and plus "+" operators. For instance, `LD+E` refers to light that has been scattered only by diffuse surfaces before reaching the camera. We will use this formalism shortly.

4.2 Path Tracing Variants

The path tracing algorithm discussed in Section 3 constructs complete light paths by randomly sampling them one vertex at a time (we refer to this as *sequential* sampling). In each iteration, it randomly chooses an additional light path vertex $\mathbf{x}_{i-1}$ using a probability density that is proportional to the (partial) weighting function $\varphi(\cdots\mathbf{x}_{i-1}\mathbf{x}_i\mathbf{x}_{i+1}\cdots)$ involving only factors that depend on the previous two vertices, i.e. $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ (this is a variant of the Markov property). The indices decrease because the algorithm constructs paths in reverse; intuitively, it searches for the trajectory of an idealized light "particle" that moves backwards in time until its emission point on the light source is found.

Path tracing performs poorly when the emission point of a light path is challenging to find, so that complete light paths are constructed with low probability. This occurs in a wide range of situations; Figure 5 shows an example where the light sources are encased, making it hard to reach them by chance. The path tracing rendering has unacceptably high variance at 32 samples per pixel.

The path space view makes it possible to construct other path tracing variants with better behavior. For instance, we can reverse the direction of the random walk and generate vertex $\mathbf{x}_{i+1}$ from $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$, which leads to a method referred to as *light tracing* or *particle tracing*. This method sends out particles from the light source (thus avoiding problems with the enclosure) and records the contribution to rendered pixels when they hit the aperture of the camera.

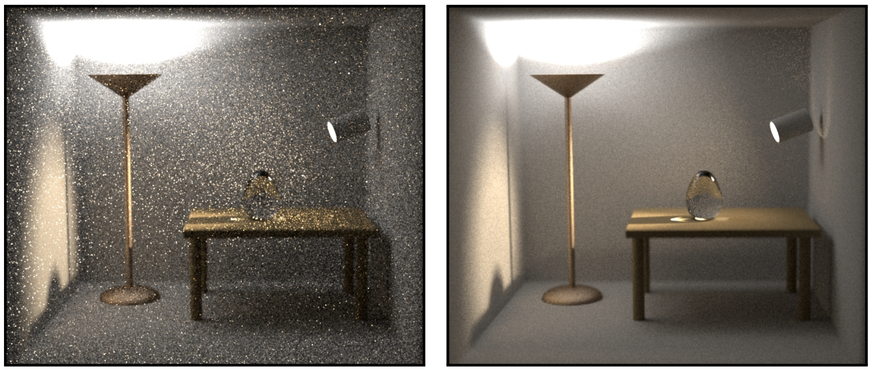

(a) Path tracer, 32 samples/pixel (b) Bidirectional path tracer, 32 samples/pixel

Fig. 5: A bidirectional path tracer finds light paths by generating partial paths starting at the camera and light sources and connecting them in every possible way. The resulting statistical estimators tend to have lower variance than unidirectional techniques. Modeled after a scene by Eric Veach.

4.2.1 Bidirectional Path Tracing (BDPT)

The bidirectional path tracing method (BDPT) [29, 17] computes radiance estimates via two separate random walks from the light sources and the camera. The resulting two partial paths are connected for every possible vertex pair, creating many complete paths of different lengths, which supplies this method with an entire family of path sampling strategies. A path with n vertices can be created in $n+1$ different ways, which is illustrated by Figure 6 for a simple path with 3 vertices (2 endpoints and 1 scattering event). The captions s and t indicate the number of sampling steps from the camera and light source. In practice, each of the strategies is usually successful at dealing with certain types of light paths, while being a poor choice for others (Figure 7).

4.2.2 Multiple Importance Sampling (MIS)

Because all strategies are defined on the same space (i.e. path space), and because each has a well-defined density function on this space, it is possible to evaluate and compare these densities to determine the most suitable strategy for sampling particular types of light paths. This is the key insight of *multiple importance sampling* (MIS) [30] which BDPT uses to combine multiple sampling strategies in a provably good way to minimize variance in the resulting rendering (bottom of Figure 7).

Suppose two statistical estimators of the pixel intensity I_j are available. These estimators can be used to generate two light paths $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$, which have path space probability densities $p_1(\bar{\mathbf{x}}_1)$ and $p_2(\bar{\mathbf{x}}_2)$, respectively. The corresponding MC

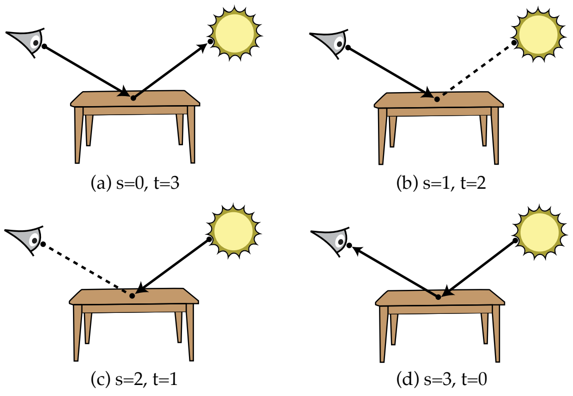


Fig. 6: The four different ways in which bidirectional path tracing can create a path with one scattering event: **(a)** Standard path tracing, **(b)** Path tracing variant: connect to sampled light source positions, **(c)** Standard light tracing, **(d)** Light tracing variant: connect to sampled camera positions. Solid lines indicate sampled rays which are intersected with the geometry, whereas dashed lines indicate deterministic connection attempts which must be validated by a visibility test.

estimates are given by

$$\langle I_j^{(1)} \rangle = \frac{\varphi(\bar{\mathbf{x}}_1)}{p_1(\bar{\mathbf{x}}_1)} \quad \text{and} \quad \langle I_j^{(2)} \rangle = \frac{\varphi(\bar{\mathbf{x}}_2)}{p_2(\bar{\mathbf{x}}_2)}.$$

To obtain a combined estimator, we could simply average these estimators, i.e.:

$$\langle I_j^{(3)} \rangle := \frac{1}{2}\big(\langle I_j^{(1)} \rangle + \langle I_j^{(2)} \rangle\big).$$

However, this is not a good idea, since the combination is affected by the variance of the worst ingredient estimator (BDPT generally uses many estimators, including ones that have very high variance). Instead, MIS combines estimators using weights that are related to the underlying sample density functions:

$$\langle I_j^{(4)} \rangle := w_1(\bar{\mathbf{x}}_1)\langle I_j^{(1)} \rangle + w_2(\bar{\mathbf{x}}_2)\langle I_j^{(2)} \rangle,$$

where

$$w_i(\bar{\mathbf{x}}) := \frac{p_i(\bar{\mathbf{x}})}{p_1(\bar{\mathbf{x}}) + p_2(\bar{\mathbf{x}})}. \tag{16}$$

While not optimal, Veach proves that no other choice of weighting functions can significantly improve on Equation (16). He goes on to propose a set of weighting heuristics that combine many estimators (i.e., more than two), and which yield perceptually better results. The combination of BDPT and MIS often yields an

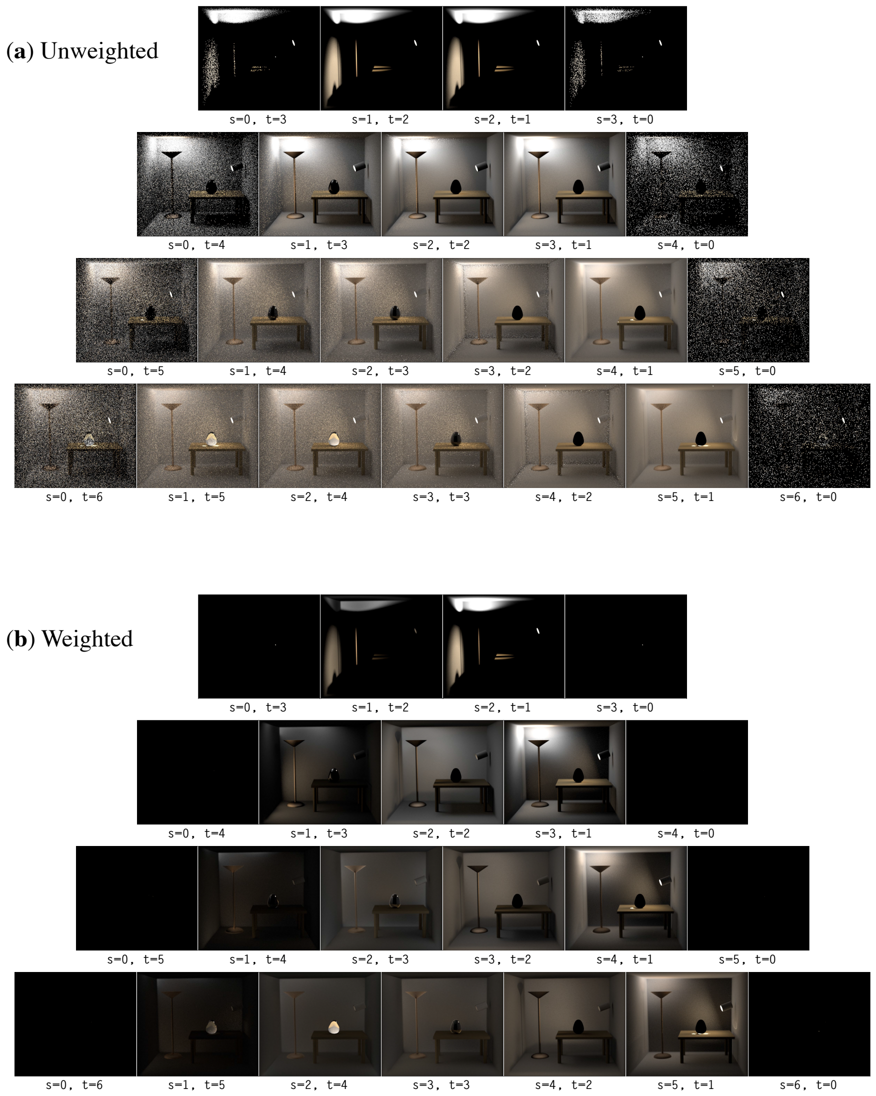


Fig. 7: The individual sampling strategies that comprise the previous BDPT rendering, both *without* (top) and *with* (bottom) multiple importance sampling. Each row corresponds to light paths of a certain length, and the top row matches the four strategies from Figure 6. Almost every strategy has deficiencies of some kind; multiple importance sampling re-weights samples to use strategies where they perform well.

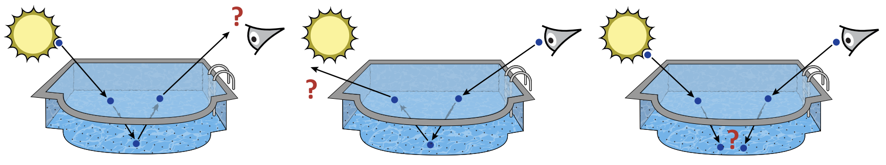
effective method that addresses many of the flaws of the path tracing algorithm. Yet, even this combination can fail in simple cases, as we will discuss next.

4.3 Limitations of Monte Carlo Path Sampling

Ultimately, all Monte Carlo path sampling techniques can be seen to compute integrals of the weighting function φ using a variety of importance sampling techniques that evaluate φ at many randomly chosen points throughout the integration domain, i.e., path space $\mathscr{P}$.

Certain input, particularly scenes containing metal, glass, or other shiny surfaces, can lead to integrals that are difficult to evaluate. Depending on the roughness of the surfaces, the integrand can take on large values over small regions of the integration domain. Surfaces of lower roughness lead to smaller and higher-valued regions, which eventually collapse to lower-dimensional sets with singular integrands as the surface roughness tends to zero. This case where certain paths cannot be sampled at all is known as the *problem of insufficient techniques* [16].

Convergence problems arise whenever high-valued regions receive too few samples. Depending on the method used, this manifests as objectionable noise or other visual artifacts in the output image that gradually disappear as the sample count N tends to infinity. However, due to the slow convergence rate of MC integration (typical error is $\mathscr{O}(N^{-0.5})$), it may not be an option to wait for the error to average out. Such situations can force users of rendering software to make unrealistic scene modifications (e.g., disabling certain light interactions), thereby compromising realism in exchange for obtaining converged-looking results within a reasonable time. Biased estimators can achieve lower errors in some situations—however, these



(a) Path tracing from the light source **(b)** Path tracing from the camera **(c)** Bidirectional path tracing

Fig. 8: Illustration of the difficulties of sequential path sampling methods when rendering `LSDSE` caustic patterns at the bottom of a swimming pool. **(a, b)**: Unidirectional techniques sample light paths by executing a random walk consisting of alternating transport and scattering steps. The only way to successfully complete a path in this manner is to randomly "hit" the light source or camera, which happens with exceedingly low probability. **(c)**: Bidirectional techniques trace paths from both sides, but in this case they cannot create a common vertex at the bottom of the pool to join the partial light paths.

methods are beyond the scope of this article, we refer the reader to Pharr et al. [24] for an overview.

Figure 8 illustrates the behavior of several path sampling methods when rendering *caustics*, which we define as light paths matching the regular expression `LS+DS*E`. They form interesting light patterns at the bottom of the swimming pool due to focusing effect of ripples in the water surface.

In Figure 8 (a), light tracing is used to emit particles proportional to the light source emission profile L_e. The highlighted path is the trajectory of a particle that encounters the water surface and refracts into the pool. The refraction is an ideal specular interaction described by Snell's law and the Fresnel equations. The diffuse concrete surface at the pool bottom then reflects the particle upwards into a direction drawn from a uniform distribution, where it is refracted once more by the water surface. Ultimately, the particle never hits the camera aperture and thus cannot contribute to the output image.

Figure 8 (b) shows the behavior of the path tracing method, which generates paths in the reverse direction but remains extremely inefficient: in order to construct a complete light path $\bar{\mathbf{x}}$ with $\varphi(\bar{\mathbf{x}}) > 0$, the path must reach the "other end" by chance, which happens with exceedingly low probability. Assuming for simplicity that rays leave the pool with a uniform distribution in Figure 8 (b), the probability of hitting the sun with an angular diameter of $\sim 0.5^\circ$ is on the order of 10^{-5}.

BDPT traces paths from both sides, but even this approach is impractical here: vertices on the water surface cannot be used to join two partial paths, since the resulting pair of incident and outgoing directions would not satisfy Snell's law. It is possible to generate two vertices at the bottom of the pool as shown in the figure, but these cannot be connected: the resulting path edge would be fully contained in a surface rather than representing transport *between* surfaces.

In this situation, biased techniques would connect the two vertices at the bottom of the pool based on a proximity criterion, which introduces systematic errors into the solution. We will only focus on unbiased techniques that do not rely on such approximations.

The main difficulty in scenes like this is that caustic paths are tightly constrained: they must start on the light source, end on the aperture, and satisfy Snell's law in two places. Sequential sampling approaches are able to satisfy all but one constraint and run into issues when there is no way to complete the majority of paths.

Paths like the one examined in Figure 8 lead to poor convergence in other settings as well; they are collectively referred to as *specular–diffuse–specular* (`SDS`) paths due to the occurrence of this sequence of interactions in their path classification. `SDS` paths occur in common situations such as a tabletop seen through a drinking glass standing on it, a bottle containing shampoo or other translucent liquid, a shop window viewed and illuminated from outside, as well as scattering inside the eye of a virtual character. Even in scenes where these paths do not cause dramatic effects, their presence can lead to excessively slow convergence in rendering algorithms that attempt to account for all transport paths. It is important to note that while the `SDS` class of paths is a well-studied example case, other classes (e.g., involving glossy interactions) can lead to many similar issues. It is desirable that rendering methods

Algorithm 2 Pseudocode of a MCMC-based rendering algorithm

```
function METROPOLIS-LIGHT-TRANSPORT
    $\bar{\mathbf{x}}_0 \leftarrow$ An initial light path
    for $i = 1$ to $N$ do
        $\bar{\mathbf{x}}'_i \leftarrow$ MUTATE($\bar{\mathbf{x}}_{i-1}$)
        $\bar{\mathbf{x}}_i \leftarrow \begin{cases} \bar{\mathbf{x}}'_i & \text{with probability } \min\left\{1, \frac{\varphi(\bar{\mathbf{x}}'_i)T(\bar{\mathbf{x}}'_i,\bar{\mathbf{x}}_{i-1})}{\varphi(\bar{\mathbf{x}}_{i-1})T(\bar{\mathbf{x}}_{i-1},\bar{\mathbf{x}}'_i)}\right\} \\ \bar{\mathbf{x}}_{i-1} & \text{otherwise} \end{cases}$
        RECORD($\bar{\mathbf{x}}_i$)
```

are robust to such situations. Correlated path sampling techniques based on MCMC offer an attractive way to approach such challenges. We review these methods in the remainder of this article.

5 Markov Chain Monte Carlo (MCMC) Rendering Techniques

In 1997, Veach and Guibas proposed an unusual rendering technique named Metropolis Light Transport [31], which applies the Metropolis-Hastings algorithm to the path space integral in Equation (14). Using correlated samples and highly specialized mutation rules, their approach enables more systematic exploration of the integration domain, avoiding many of the problems encountered by methods based on standard Monte Carlo and sequential path sampling.

Later, Kelemen et al. [14] showed that a much simpler approach can be used to combine MCMC sampling with existing MC rendering algorithms, making it possible to side-step the difficulties of the former method. The downside of their approach is the reduced flexibility in designing custom mutation rules. An extension by Hachisuka et al. [7] further improves the efficiency of this method.

Considerable research has built on these two approaches, including extensions to participating media [23], combinations of MCMC and BDPT [7], specialized techniques for specular [11] and glossy [13] materials, gradient-domain rendering [18, 19], and MCMC variants which perform a localized non-ergodic exploration of path space [3].

In this section, we provide an overview of the initial three methods, starting first with the Primary Sample Space approach by Kelemen et al., followed the extension by Hachisuka et al., and finally the Metropolis Light Transport algorithm by Veach and Guibas. All variants are based on a regular MCMC iteration shown in Algorithm 2. Starting with an initial light path $\bar{\mathbf{x}}_0$, the methods simulate N steps of a Markov Chain. In each step, a mutation is applied to the path $\bar{\mathbf{x}}_{i-1}$ to obtain a proposal path $\bar{\mathbf{x}}'_i$, where it is assumed that the proposal density is known and given by $T(\bar{\mathbf{x}}_{i-1}, \bar{\mathbf{x}}'_i)$. After a standard Metropolis-Hastings acceptance/rejection step, the algorithm invokes the function RECORD($\bar{\mathbf{x}}_i$), which first determines the pixel associated with the current iteration's light path $\mathbf{x}_i$ and then increases its brightness by a fixed amount.

These MCMC methods all sample light paths proportional to the amount they contribute to the pixels of the final rendering; by increasing the pixel brightness in this way during each iteration, these methods effectively compute a 2D histogram of the marginal distribution of φ over pixel coordinates. This is exactly the image to be rendered up to a global scale factor, which can be recovered using a traditional MC sampling technique such as BDPT. The main difference among these algorithms is the underlying state space, as well as the employed set of mutation rules.

5.1 Primary Sample Space Metropolis Light Transport (PSSMLT)

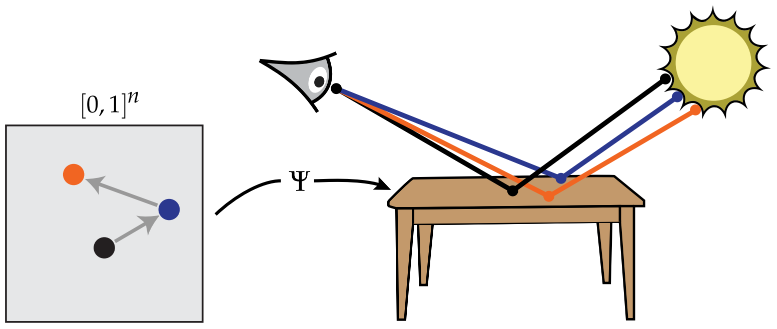
Primary Sample Space Metropolis Light Transport (PSSMLT) [14] combines traditional MC sampling techniques with a MCMC iteration. The approach is very flexible and can also be applied to integration problems outside of computer graphics. PSSMLT always operates on top of an existing MC sampling technique; we assume for simplicity that path tracing is used, but many other techniques are also admissible. The details of this method are easiest to explain from a implementation-centric viewpoint.

Recall the path tracing pseudo-code shown earlier in Algorithm 1. Lines 1 and 2 performed random sampling steps, but the rest of the procedure was fully deterministic. In practice, the first two lines are often realized using a pseudorandom number generator such as Mersenne Twister [20] or a suitable quasi-Monte Carlo scheme [6], potentially using the inversion method or a similar technique to warp uniform variates to desired distributions as needed. For more details, we refer the reader to a tutorial by Keller [15].

Let us consider a small adjustment to the implementation of this method: instead of generating univariate samples during the recursive sampling steps, we can also generate them ahead of time and supply them to the implementation as an additional argument, in which case the algorithm can be interpreted as a fully deterministic function of its (random or pseudorandom) arguments. Suppose that we knew (by some way) that the maximum number of required random variates was equal to n, and that the main computation was thus implemented by a function with signature $\Psi : [0,1]^n \to \mathbb{R}$, which maps a vector of univariate samples to a pixel intensity estimate. By taking many estimates and averaging them to obtain a converged pixel intensity, path tracing is effectively integrating the estimator over a n-dimensional unit hypercube of "random numbers" denoted as *primary sample space*:

$$I_j = \int_{[0,1]^n} \Psi(\xi)\,\mathrm{d}\xi. \tag{17}$$

The key idea of PSSMLT is to compute Equation (17) using MCMC integration on primary sample space, which leads to a trivial implementation, as all complications involving light paths and other rendering-specific details are encapsulated in the "black box" mapping Ψ (Figure 9).



(a) Primary sample space view **(b)** Path space view

Fig. 9: Primary Sample Space MLT performs mutations in an abstract random number space. A deterministic mapping Ψ induces corresponding mutations in path space.

One missing detail is that the primary sample space dimension n is unknown ahead of time. This can be solved by starting with a low-dimensional integral and extending the dimension on demand when additional samples are requested by Ψ.

PSSMLT uses two types of MUTATE functions. The first is an independence sampler, i.e., it forgets the current state and switches to a new set of pseudorandom variates. This is needed to ensure that the Markov Chain is ergodic. The second is a local (e.g. Gaussian or similar) proposal centered around a current state $\xi_i \in [0,1]^n$. Both are symmetric so that the proposal density T cancels in the acceptance ratio (Line 5 in Algorithm 2).

PSSMLT uses independent proposals to find important light paths that cannot be reached using local proposals. When it finds one, local proposals are used to explore neighboring light paths which amortizes the cost of the search. This can significantly improve convergence in many challenging situations and is an important advantage of MCMC methods in general when compared to MC integration.

Another advantage of PSSMLT is that it explores light paths through a black box mapping Ψ that already makes internal use of sophisticated importance sampling techniques for light paths, which in turn leads to an easier integration problem in primary sample space. The main disadvantage of this method is that its interaction with Ψ is limited to a stream of pseudorandom numbers. It has no direct knowledge of the generated light paths, which prevents the design of more efficient mutation rules based on the underlying physics.

5.2 *Multiplexed Metropolis Light Transport (MMLT)*

PSSMLT is commonly implemented in conjunction with the BDPT technique: in this setting, the rendering algorithm generates paths using a large set of BDPT connection

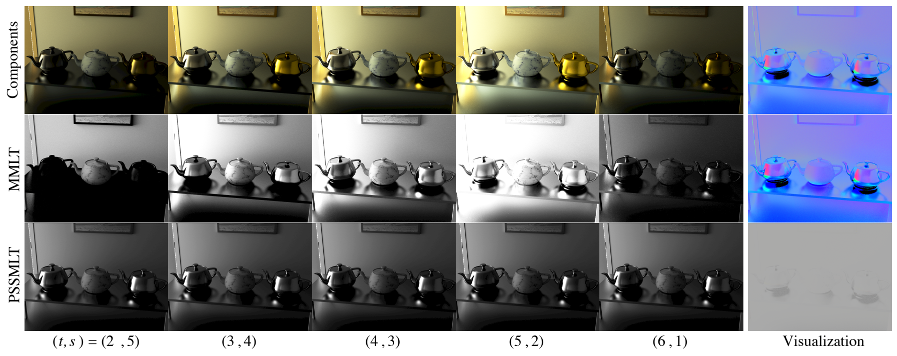


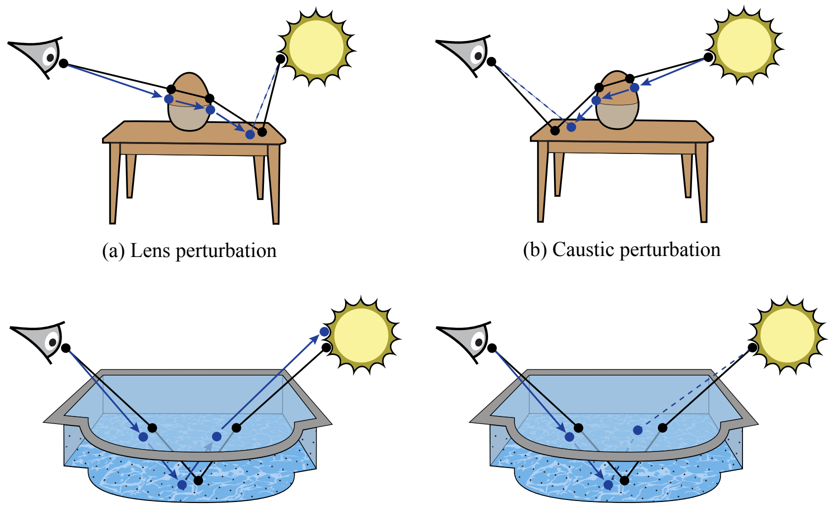
Fig. 10: Analysis of the Multiplexed MLT (MMLT) technique [7] (used with permission): the top row shows weighted contributions from different BDPT strategies in a scene with challenging indirect illumination [18, 28]. The intensities in the middle row visualize the time spent on each strategy using the MMLT technique: they are roughly proportional to the weighted contribution in the first row. The rightmost column visualizes the dominant strategies (3,4), (4, 3), and (5, 2) using RGB colors. PSSMLT (third row) cannot target samples in this way and thus produces almost uniform coverage.

strategies and then re-weights them using MIS. In most cases, only a subset of the strategies is truly effective, and MIS will consequently assign a large weight to this subset. One issue with the combination of BDPT and PSSMLT is that the algorithm still spends a considerable portion of its time generating connections with strategies that have low weights and thus contribute very little to the rendered image. Hachisuka et al. [7] recently presented an extension of PSSMLT named *Multiplexed Metropolis Light Transport* (MMLT) to address this problem.

They propose a simple but effective modification to the inner BDPT sampler; the outer Metropolis-Hastings iteration remains unchanged: instead of generating a sample from all BDPT connection strategies, the algorithm (pseudo-)randomly chooses a single strategy and returns its contribution scaled by the inverse discrete probability of the choice. This (pseudo-)random sample is treated in the same way as other sampling operations in PSSMLT and exposed as an additional state dimension that can be mutated using small or large steps. The practical consequence is that the Markov Chain will tend to spend more computation on effective strategies, which further improves the statistical efficiency of the underlying estimator (Figure 10).

5.3 Path Space Metropolis Light Transport (MLT)

Path Space Metropolis Light transport, or simply Metropolis Light transport (MLT) [31] was the first application of MCMC to the problem of light transport.


(a) Lens perturbation (b) Caustic perturbation

(c) Multi-chain perturbation (d) Manifold perturbation

Fig. 11: MLT operates on top of path space, which permits the use of a variety of mutation rules that are motivated by important physical scattering effects. The top row illustrates ones that are useful when rendering a scene involving a glass object on top of a diffuse table. The bottom row is the swimming pool example from Figure 8. In each example, the original path is black, and the proposal is highlighted in blue.

Doucet et al. [5] proposed a related method in applied mathematics, which focuses on a more general class of integral equations.

The main difference as compared to PSSMLT is that MLT operates directly on path space and does not use a black-box mapping Ψ. Its mutation rules are considerably more involved than those of PSSMLT, but this also provides substantial freedom to design custom rules that are well-suited for rendering specific physical effects. MLT distinguishes between *mutations* that change the structure of the path and *perturbations* that move the vertices by small distances while preserving the path structure, both using the building blocks of bidirectional path tracing to sample paths. One of the following operations is randomly selected in each iteration:

1. **Bidirectional mutation**: This mutation replaces a segment of an existing path with a new segment (possibly of different length) generated by a BDPT-like sampling strategy. This rule generally has a low acceptance ratio but it is essential to guarantee ergodicity of the resulting Markov Chain.
2. **Lens subpath mutation**: The lens subpath mutation is similar to the previous mutation but only replaces the *lens subpath*, which is defined as the trailing portion of the light path matching the regular expression `[^S]S*E`.
3. **Lens perturbation**: This transition rule shown in Figure 11a only perturbs the lens subpath rather than regenerating it from scratch. In the example, it slightly

rotates the outgoing ray at the camera and propagates it until the first non-specular material is encountered. It then attempts to create a connection (dashed line) to the unchanged remainder of the path.

4. **Caustic perturbation**: The caustic perturbation (Figure 11b) works just like the lens perturbation, except that it proceeds in reverse starting at the light source. It is well-suited for rendering caustics that are directly observed by the camera.
5. **Multi-chain perturbation**: This transition rule (Figure 11c) is used when there are multiple separated specular interactions, e.g., in the swimming pool example encountered before. After an initial lens perturbation, a cascade of additional perturbations follows until a connection to the remainder of the path can finally be established.

The main downside of MLT is the severe effort needed to implement this method: several of the mutation and perturbation rules (including their associated proposal densities) are challenging to reproduce. Another problem is that a wide range of different light paths generally contribute to the output image. The MLT perturbations are designed to deal with specific types of light paths, but it can be difficult to foresee every kind in order to craft a suitable set of perturbation rules. In practice, the included set is insufficient.

6 Specular Manifolds and Manifold Exploration (ME)

In this section, we discuss the principles of *Manifold Exploration* (ME) [11], which leads to the manifold perturbation (Figure 11d). This perturbation provides local exploration for large classes of different path types and subsumes MLT's original set of perturbations. We begin with a discussion of the concept of a *specular manifold*. When a scene contains ideal specular materials, these materials require certain physical laws to be satisfied (e.g. Snell's law or the law of reflection). Mathematically, these act like constraint equations that remove some dimensions of the space of light paths, leaving behind a lower-dimensional manifold embedded in path space.

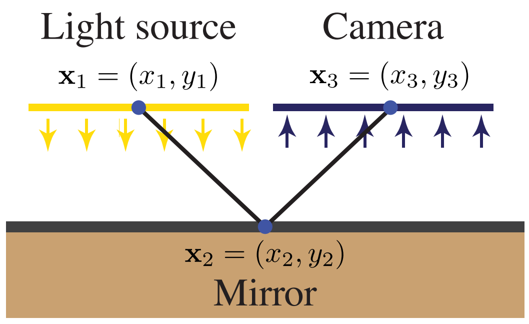

Light source
Camera
$\mathbf{x}_1 = (x_1, y_1)$
$\mathbf{x}_3 = (x_3, y_3)$
$\mathbf{x}_2 = (x_2, y_2)$
Mirror

Fig. 12: A motivating example in two dimensions: specular reflection in a mirror

We illustrate this using a simple example in two dimensions, in which a camera observes a planar light source through an opposing mirror (Figure 12). We will refer to a light path joining two endpoints through a sequence of k ideal specular scattering events as a *specular chain* of length k. A specular chain of length 1 from the light source to the camera is shown in the figure.

Reflections in the mirror must satisfy the law of specular reflection. Assuming that the space of all specular chains in this simple scene can be parameterized using the horizontal coordinates x_1, x_2, and x_3, it states that

$$x_2 = \frac{x_1 + x_3}{2}, \tag{18}$$

i.e., the x coordinate of the second vertex must be exactly half-way between the endpoints. Note that this equation can also be understood as the implicit definition of a plane in $\mathbb{R}^3$ $(x_1 - 2x_2 + x_3 = 0)$.

When interpreting the set of all candidate light paths as a three-dimensional space $\mathscr{P}_3$ of coordinate tuples (x_1, x_2, x_3), this constraint then states that the subset of relevant paths has one dimension less and is given by the intersection of $\mathscr{P}_3$ and the plane equation (18). With this extra knowledge, it is now easy to sample valid specular chains, e.g. by generating x_1 and x_3 and solving for x_2.

Given general non-planar shapes, the problem becomes considerably harder, since the equations that have to be satisfied are nonlinear and may admit many solutions. Prior work has led to algorithms that can find solutions even in such cases [21, 33] but these methods are closely tied to the representation of the underlying geometry, and they become infeasible for specular chains with lengths greater than one. Like these works, ME finds valid specular chains—but because it does so within the *neighborhood* of a given path, it avoids the complexities of a full global search and does not share these limitations.

ME is also related to the analysis of reflection geometry presented by Chen and Arvo [2], who derived second-order expansion of the neighborhood of a path. The main difference is that ME solves for paths exactly and is used as part of an unbiased MCMC rendering algorithm.

6.1 Integrals over Specular Manifolds

Let us return to our previous example of the swimming pool involving the family of light paths `LSDSE`. These paths belong to the $\mathscr{P}_5$ component of the path space $\mathscr{P}$ (Equation 12), which is a 10-dimensional space with two dimensions for each surface position. As we will see shortly, the paths that contribute have to satisfy two constraint equations involving unit directions in $\mathbb{R}^3$ (which each have 2 degrees of freedom). This constrains a total of four dimensions of the path, meaning that all contributing paths lie on a manifold $\mathscr{S}$ of dimension 6 embedded in $\mathscr{P}_5$.

The corresponding integral equation (13) is more naturally expressed as an integral over this *specular manifold* $\mathscr{S}$, rather than as an integral over the entire path space:

$$\int_{\mathscr{S}} \varphi(\mathbf{x}_1 \cdots \mathbf{x}_5)\, \mathrm{d}A(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_5).$$

Note the absence of the specular vertices $\mathbf{x}_2$ and $\mathbf{x}_4$ in the integral's area product measure. The contribution function φ still has the same form: a product of terms corresponding to vertices and edges of the path. However, singular reflection functions at specular vertices are replaced with (unitless) specular reflectance values, and the geometric terms are replaced by generalized geometric terms over specular chains that we will denote $G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3)$ and $G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_4 \leftrightarrow \mathbf{x}_5)$.

The standard geometric term $G(\mathbf{x} \leftrightarrow \mathbf{y})$ for a non-specular edge computes the area ratio of an (infinitesimally) small surface patch at one vertex and its projection onto projected solid angles as seen from the other vertex. The generalized geometry factor is defined analogously: the ratio 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.

6.2 *Constraints for Reflection and Refraction*

Equation (18) introduced a simple specular reflection constraint for axis-aligned geometry in two dimensions. This constraint easily generalizes to arbitrary geometry in three dimensions and to both specular reflection and refraction.

Recall the law of specular reflection, which states that incident and outgoing directions make the same angle with the surface normal. Furthermore, all three vectors must be coplanar (Figure 13). We use an equivalent reformulation of this law, which states that the *half direction vector* of the incident and outgoing direction ω_i and ω_o, defined as

$$\mathbf{h}(\omega_i, \omega_o) := \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}, \tag{19}$$

is equal to the surface normal, i.e., $\mathbf{h}(\omega_i, \omega_o) = \mathbf{n}$. In the case of refraction, the relationship of these directions is explained by Snell's law. Using a generalized definition of the half direction vector which includes weighting by the incident and

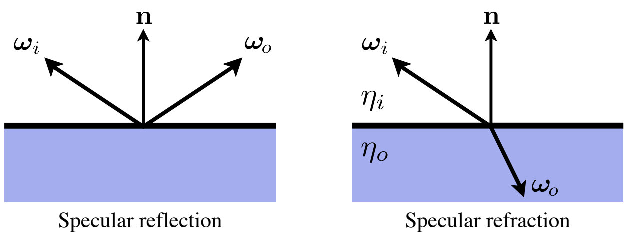


Fig. 13: In-plane view of the surface normal $\mathbf{n}$ and incident and outgoing directions ω_i and ω_o at a surface marking a transition between indices of refraction η_i and η_o.

outgoing indices of refraction [32]; i.e.,

$$\mathbf{h}(\omega_i, \omega_o) := \frac{\eta_i \omega_i + \eta_o \omega_o}{\|\eta_i \omega_i + \eta_o \omega_o\|}, \tag{20}$$

we are able to use a single constraint $\mathbf{h}(\omega_i, \omega_o) = \pm\mathbf{n}$ which subsumes both Snell's law and the law of specular reflection (in which case η_i equals η_o). Each specular vertex $\mathbf{x}_i$ of a path $\bar{\mathbf{x}}$ must satisfy this generalized constraint involving its own position and the positions of the preceding and following vertices. Note that this constraint involves unit vectors with only *two* degrees of freedom. We can project (20) onto a two-dimensional subspace to reflect its dimensionality:

$$\mathbf{c}_i(\bar{\mathbf{x}}) = T(\mathbf{x}_i)^T \mathbf{h}(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}), \tag{21}$$

The functions $\mathbf{c}_i : \mathscr{P} \to \mathbb{R}^2$ compute the generalized half-vector at vertex $\mathbf{x}_i$ and project it onto the tangent space of the underlying scene geometry at this position, which is spanned by the columns of the matrix $T(\mathbf{x}_i) \in \mathbb{R}^{3\times 2}$; the resulting 2-vector is zero when $\mathbf{h}(\omega_i, \omega_o)$ is parallel to the normal. Then the specular manifold is simply the set

$$\mathscr{S} = \{\bar{\mathbf{x}} \in \mathscr{P} \mid \mathbf{c}_i(\bar{\mathbf{x}}) = \mathbf{0} \text{ if vertex } \mathbf{x}_i \text{ is specular}\}. \tag{22}$$

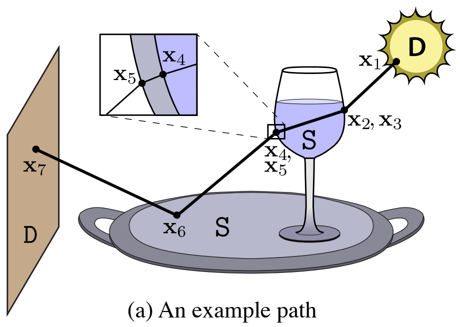
6.3 Local Manifold Geometry

The complex nonlinear behavior of $\mathscr{S}$ severely limits our ability to reason about its geometric structure globally. In this section, we therefore focus on *local* properties, leading to an explicit expression for the tangent space at any point on the manifold. This constitutes the key geometric information needed to construct a numerical procedure that is able to move between points on the manifold.

For simplicity, let us restrict ourselves to the case of a single specular chain $\bar{\mathbf{x}} = \mathbf{x}_1 \cdots \mathbf{x}_k$ with $k-2$ specular vertices and non-specular endpoints $\mathbf{x}_1$ and $\mathbf{x}_k$, matching the path regular expression `DS+D`. This suffices to cover most cases by separate application to each specular chain along a path. To analyze the geometry locally, we require a point in $\mathscr{S}$, i.e., a light path $\bar{\mathbf{x}}$ satisfying all specular constraints, to be given.

We assume that local parameterizations of the surfaces in the scene on small neighborhoods around every vertex are provided via functions $\hat{\mathbf{x}}_i(u_i, v_i) : \mathbb{R}^2 \to \mathscr{M}$, where $\hat{\mathbf{x}}_i(0,0) = \mathbf{x}_i$. We can then express the constraints $\mathbf{c}_i$ in terms of these local coordinates and stack them on top of each other to create a new function $\hat{\mathbf{c}}$ with signature $\hat{\mathbf{c}} : \mathbb{R}^{2k} \to \mathbb{R}^{2k-4}$, which maps $2k$ local coordinate values to $2k-4 = 2(k-2)$ projected half direction vector coordinates—two for each of the specular vertices of the chain. The set

$$\mathscr{S}_{\text{loc}} = \left\{(u_1, v_1, \ldots, u_k, v_k) \in \mathbb{R}^{2k} \;\middle|\; \hat{\mathbf{c}}(\ldots) = \mathbf{0}\right\} \tag{23}$$

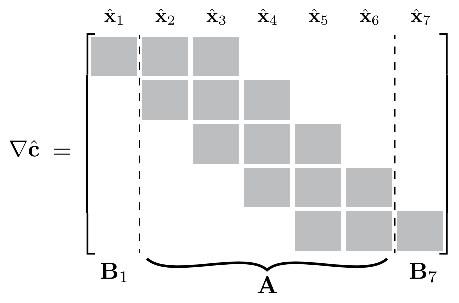


(a) An example path

$$\hat{\mathbf{c}}(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_7) = \mathbf{0} \ \text{ where}$$

$$\hat{\mathbf{c}} = \begin{bmatrix} \hat{\mathbf{c}}_2(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_7) \\ \vdots \\ \hat{\mathbf{c}}_6(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_7) \end{bmatrix}$$

(b) Associated constraints



(c) Constraint Jacobian

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

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

(d) Tangent space

Fig. 14: The linear system used to compute the tangent space and its interpretation as a derivative of a specular chain.

then describes the (four-dimensional) specular manifold in terms of local coordinates around the path $\bar{\mathbf{x}}$, which is identified with the origin. Under the assumption that the Jacobian of $\hat{\mathbf{c}}$ has full rank (more on this shortly), the Implicit Function Theorem [26] states that the implicitly defined manifold (23) can be converted into the (explicit) graph of a function $\mathbf{q} : \mathbb{R}^4 \to \mathbb{R}^{2k-4}$ on an epsilon ball $\mathscr{B}_4(\varepsilon)$ around the origin. Different functions $\mathbf{q}$ are possible—in our case, the most useful variant determines the positions of all the specular vertices from the positions of the non-specular endpoints, i.e.

$$\mathscr{S}'_{\text{loc}} = \big\{(u_1, v_1, \mathbf{q}(u_1, v_1, u_k, v_k), u_k, v_k) \;\big|\; (u_1, v_1, u_k, v_k) \in \mathscr{B}_4(\varepsilon)\big\}. \tag{24}$$

Unfortunately, the theorem does not specify *how* to compute $\mathbf{q}$—it only guarantees the *existence* of such a function. It does, however, provide an explicit expression for the derivative of $\mathbf{q}$, which contains all information we need to compute a basis for the tangent space at the path $\bar{\mathbf{x}}$, which corresponds to the origin in local coordinates. This involves the Jacobian of the constraint function $\nabla\hat{\mathbf{c}}(\mathbf{0})$, which is a matrix of $k-2$ by k 2-by-2 blocks with a block tridiagonal structure (Figure 14).

If we block the derivative $\nabla\hat{\mathbf{c}}$, as shown in the figure, into 2-column matrices $\mathbf{B}_1$ and $\mathbf{B}_k$ for the first and last vertices and a square matrix $\mathbf{A}$ for the specular chain, the tangent space to the manifold in local coordinates is

$$T_{\mathscr{S}}(\bar{\mathbf{x}}) = -\mathbf{A}^{-1}\left[\mathbf{B}_1\ \mathbf{B}_k\right]. \tag{25}$$

This matrix is $k-2$ by 2 blocks in size, and each block represents the derivative of one vertex with respect to one endpoint.

This construction computes tangents with respect to a graph parameterization of the manifold, which is guaranteed to exist for a suitable choice of independent variables. Because we always use the endpoint vertices for this purpose, difficulties arise when one of the endpoints is located exactly at the fold of a caustic wavefront, in which case $\nabla\hat{\mathbf{c}}$ becomes rank-deficient and $\mathbf{A}$ fails to be invertible. This happens rarely in practice and is not a problem for our method, which allows for occasional parameterization failures. In other contexts where this is not acceptable, the chain could be parameterized by a different pair of vertices when a non-invertible matrix is detected.

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

6.4 Walking on the Specular Manifold

In practice, we always keep one endpoint fixed (e.g., $\mathbf{x}_1$), while parameterizing the remaining two-dimensional set. Figure 15 shows a conceptual sketch of the manifold of a specular chain that is parameterized by the last vertex $\mathbf{x}_k$. This vertex is initially located at $\mathbf{x}_k^{\text{start}}$, and we search for a valid configuration where it is at position $\mathbf{x}_k^{\text{target}}$. The derivation in Section 6.3 provides a way of extrapolating the necessary change of $\mathbf{x}_2,\ldots,\mathbf{x}_{k-1}$ to first order, but this is not enough: an expansion, no matter to what order, will generally not be able to find a valid path that is located on $\mathscr{S}$.

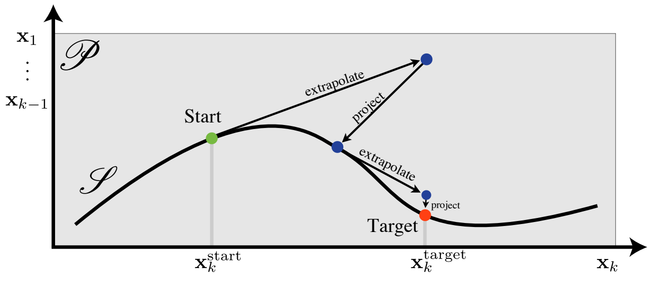


Fig. 15: Manifold walks use a Newton-like iteration to locally parameterize the specular manifold. The extrapolation operation takes first-order steps based on the local manifold tangents, which are subsequently projected back onto the manifold.

To address this issue, we combine the extrapolation with a simple projection operation, which maps approximate paths back onto $\mathscr{S}$ by intersecting the extrapolated ray $\mathbf{x}_1 \to \mathbf{x}_2$ with the scene geometry and using the appropriate laws of reflection and refraction to compute the remaining vertex locations. The combination of extrapolation and projection behaves like Newton's method, exhibiting quadratic convergence near the solution; details on this iteration can be found in the original paper [11].

Figure 16 shows a sketch of how manifold walks can be used in a MLT-like iteration: a proposal begins to modify a light path by perturbing the outgoing direction at vertex $\mathbf{x}_a$. Propagating this direction through a specular reflection leads to a modified position $\mathbf{x}'_b$ on a diffuse surface. To complete the partial path, it is necessary to find a specular chain connecting $\mathbf{x}'_b$ to the light source. Here, we can simply apply a manifold walk to the existing specular chain $\mathbf{x}_b \cdots \mathbf{x}_c$ to solve for an updated configuration $\mathbf{x}'_b \cdots \mathbf{x}_c$. The key observation is that MCMC explores the space of light paths using localized steps, which is a perfect match for the local parameterization of the path manifold provided by Manifold Exploration.

6.5 Results

Figures 17 and 18 show the comparisons of several MCMC rendering techniques for an 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 rendering time was allotted to each technique; the results are intentionally unconverged to permit a visual analysis of the convergence behavior. By reasoning about the geometry of the specular and offset specular manifolds for the paths it encounters, the ME perturbation strategy is more successful at rendering certain 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.

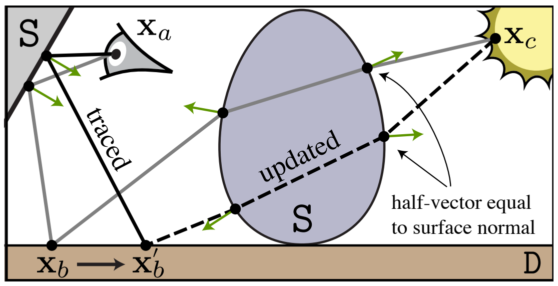
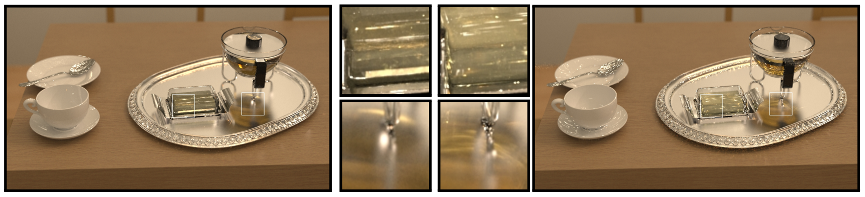
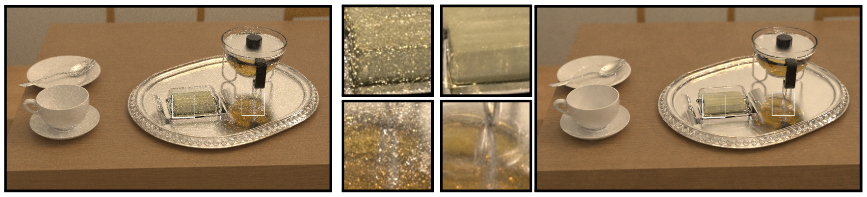

S
$\mathbf{x}_a$
traced
updated
S
$\mathbf{x}_c$
half-vector equal to surface normal
$\mathbf{x}_b \longrightarrow \mathbf{x}'_b$
D

Fig. 16: Example of a manifold-based path perturbation.

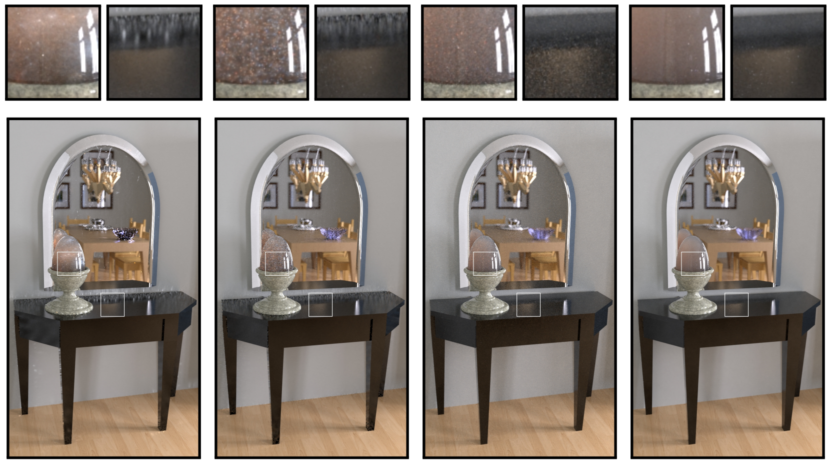


(a) MLT [28] (b) ERPT [3]



(c) PSSMLT [14] (d) ME [11]

Fig. 17: This interior scene shows chinaware, a teapot containing an absorbing medium, and a butter dish on a glossy silver tray. Illumination comes from a complex chandelier with glass-enclosed bulbs. Prior methods have difficulty in finding and exploring relevant light paths, which causes noise and other convergence artifacts. Equal-time renderings on an eight-core Intel Xeon X5570 machine at 1280×720 pixels in 1 hour.



(a) MLT [28] (b) ERPT [3] (c) PSSMLT [14] (d) ME [11]

Fig. 18: This view of a different part of the room, now lit through windows using a spherical environment map surrounding the scene, contains a scattering medium inside the glass egg. Equal-time renderings at 720×1280 pixels in 1 hour.

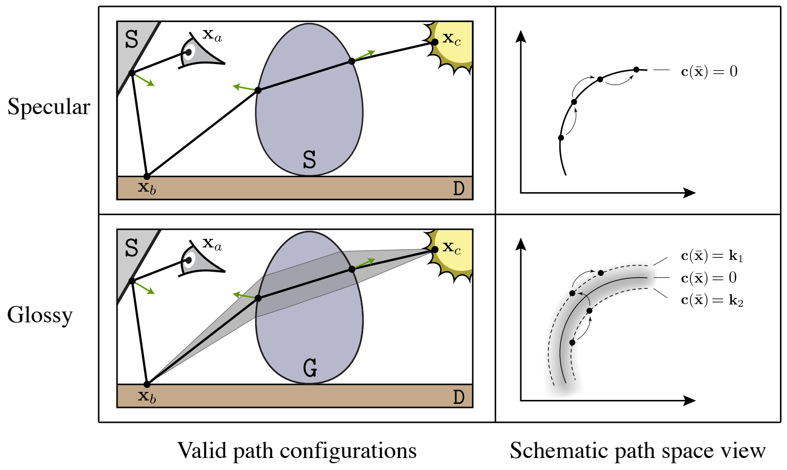

Specular
Glossy
S
$\mathbf{x}_a$
$\mathbf{x}_b$
$\mathbf{x}_c$
S
G
D
$\mathbf{c}(\bar{\mathbf{x}}) = 0$
$\mathbf{c}(\bar{\mathbf{x}}) = \mathbf{k}_1$
$\mathbf{c}(\bar{\mathbf{x}}) = 0$
$\mathbf{c}(\bar{\mathbf{x}}) = \mathbf{k}_2$
Valid path configurations
Schematic path space view

Fig. 19: Sketch of the generalization of Manifold Exploration to glossy materials.

7 Perturbation Rules for Glossy Transport

Realistic scenes contain a diverse set of materials and are usually not restricted to specular or diffuse BSDFs. It is important for the used rendering method to generalize to such cases. All derivations thus far focused on ideal specular materials, but it is possible to extend manifold walks to glossy materials as well. Jakob and Marschner proposed a simple generalization of ME, which works for moderately rough materials, and Kaplanyan et al. [13] recently developed a natural constraint representation of light paths. They proposed a novel half vector-based perturbation rule as well as numerous enhancements including better tolerance to non-smooth geometry and sample stratification in image space based on a frequency analysis of the scattering operator. We provide a high level overview of both approaches here.

7.1 Glossy Materials in the Manifold Perturbation

Figure 19 shows a sketch of this generalization. In the ideal specular case, there is a single specular chain (or discrete set) connecting $\mathbf{x}_b$ and $\mathbf{x}_c$ (top left), and all energy is concentrated on a lower-dimensional specular manifold defined by $\mathbf{c}(\bar{\mathbf{x}}) = \mathbf{0}$ (top right). In the glossy case, there is a continuous family of chains connecting $\mathbf{x}_b$ and $\mathbf{x}_c$ (bottom left), and the space of light paths has its energy concentrated in a thin “band” *near* the specular manifold. The key idea of how ME handles glossy materials is to take steps along a family of parallel offset manifolds $\mathbf{c}(\bar{\mathbf{x}}) = \mathbf{k}$ (bottom right) so that path space near the specular manifold can be explored without stepping out of this thin band of near-specular light transport.

7.2 The Natural Constraint Formulation

The method by Kaplanyan et al. [13] takes a different approach to explore glossy transport paths (Figure 20): instead of parameterizing a glossy chain by fixing its half vectors and moving the chain endpoints, their method parameterizes complete paths starting at the light source and ending at the camera. The underlying manifold walk keeps the path endpoints fixed and computes a nearby light path as a function of its half vectors. The set of all half vectors along a path can be interpreted as a type of generalized coordinate system for light paths: its dimension equals the path's degrees of freedom, while capturing the relevant constraints (reflection and refraction) in a convenient explicit form. For this reason, the resulting parameterization is referred to as the *natural constraint representation*, and the method is called *half vector space light transport* (HSLT); loosely speaking, its perturbation can be seen to explore "orthogonal" directions as compared to the "parallel" manifold walks of ME.

The underlying approach is motivated by the following interesting observation: when parameterizing light paths in terms of their half vectors, the influence of material terms on the integrand φ approximately decouples (Figure 21). The reason for this effect is that the dominant terms in glossy reflectance models (which are factors of φ) depend on the angle between the half vector and the surface normal. The

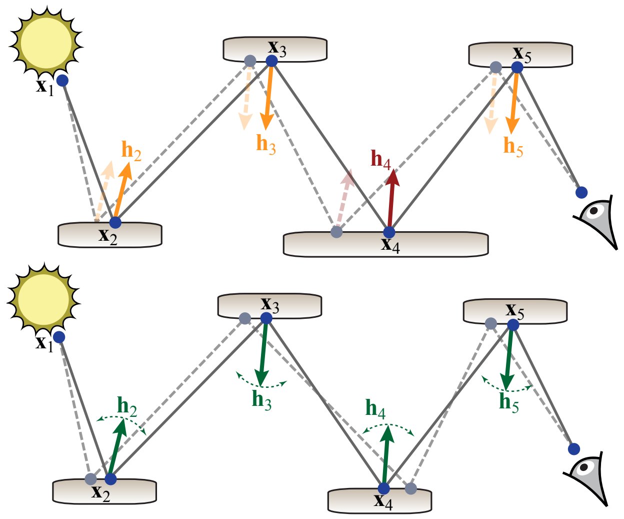


Fig. 20: In the above example, ME (top) constrains the half vectors of two glossy chains $\mathbf{x}_1 \cdots \mathbf{x}_4$ and $\mathbf{x}_4 \cdots \mathbf{x}_6$ and solves for an updated configuration after perturbing the position of $\mathbf{x}_4$. HSLT (bottom) instead adjusts all half vectors at once and solves for suitable vertex positions with this configuration. This proposal is effective for importance sampling the material terms and leads to superior convergence when dealing with transport between glossy surfaces. Based on a figure by Kaplanyan et al. [13] (used with permission).

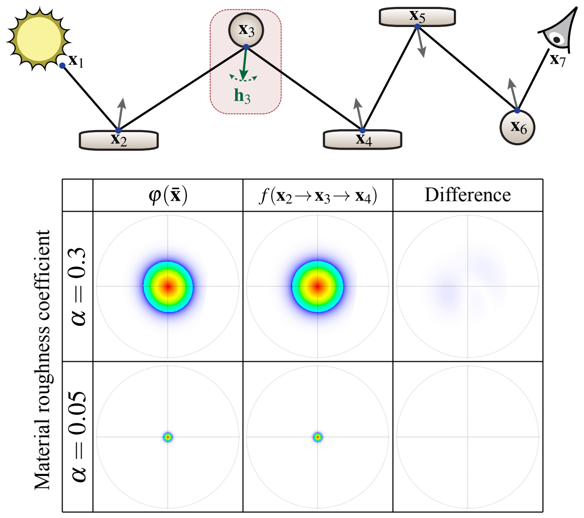


Fig. 21: The natural constraint formulation [13] is a parameterization of path space in the half vector domain. It has the interesting property of approximately decoupling the influence of the individual scattering events on φ. The figure shows a complex path where the half vector $\mathbf{h}_3$ is perturbed at vertex $\mathbf{x}_3$. The first column shows a false-color plot of φ over the resulting paths for different values of $\mathbf{h}_3$ and two roughness values. The second column shows a plot of the BSDF value at this vertex, which is approximately proportional to φ. Based on a figure by Kaplanyan et al. [13] (used with permission).

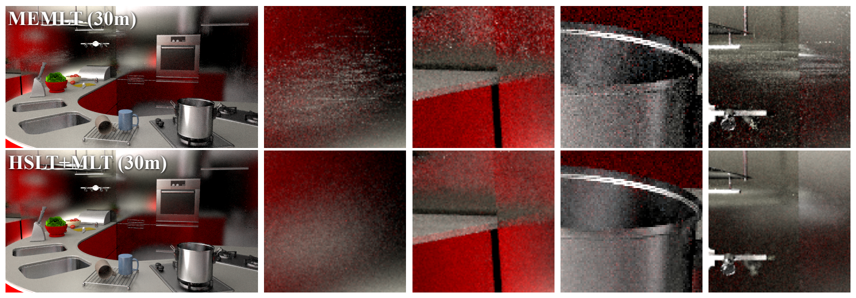


Fig. 22: Equal-time rendering of an interior kitchen scene with many glossy reflections. Based on a figure by Kaplanyan et al. [13] (used with permission).

change of variables from path space to the half vector domain furthermore cancels out the geometry terms G, leading to additional simplifications. As a consequence, this parameterization turns φ into a much simpler function resembling a separate Gaussian in each half vector dimension, which is related to the roughness of the associated surface. Kaplanyan et al. also demonstrate how frequency-space information about the scattering operator can be used to better spread out samples in image space, which is important to accelerate convergence of the histogram generation method that creates the final rendering.

Figure 22 shows a rendering comparison of a kitchen scene rendered by ME and HSLT, where most of the illumination is due to caustics paths involving a reflection by the glossy floor. After 30 minutes, the ME rendering is noticeably less converged and suffers from stripe artifacts, which are not present in the HSLT result.

8 Conclusion

This article presented an overview of the physics underlying light transport simulations in computer graphics. After introducing relevant physical quantities and the main energy balance equation, we showed how to compute approximate solutions using a simple Monte Carlo estimator. Following this, we introduced the concept of path space and examined the relation of path tracing, light tracing, and bidirectional path tracing—including their behavior given challenging input that causes these methods to become impracticably slow. The second part of this article reviewed several MCMC methods that compute path space integrals using proposal distributions defined on sets of light paths. To efficiently explore light paths involving specular materials, we showed how to implicitly define and locally parameterize the associated paths using a root-finding iteration. Finally, we reviewed recent work that aims to generalize this approach to glossy scattering interactions. Most of the methods that were discussed are implemented in the Mitsuba renderer [9], which is a research-oriented open source rendering framework.

MCMC methods in rendering still suffer from issues that limit their usefulness in certain situations. Most importantly, they require an initialization or mutation rule that provides well distributed seed paths to the perturbations, as they can only explore connected components of path space. Bidirectional Path Tracing and the Bidirectional Mutation are reasonably effective but run into issues when there are many disconnected components of path space. This becomes increasingly problematic as their number increases. Ultimately, as the number of disconnected components 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.

Furthermore, the all perturbations rules made assumptions about specific path configurations or material properties, which limits their benefits when rendering scenes that contain a wide range of material types. To efficiently deal with light paths involving arbitrary materials, camera models, and light sources, a fundamentally different construction will be needed.

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 and an ETH/Marie Curie fellowship. The author is indebted to Olesya Jakob, who crafted several of the example scenes in this article.

References

1. Arvo, J.R.: Analytic methods for simulated light transport. Ph.D. thesis, Yale University (1995)
2. Chen, M., Arvo, J.: Theory and application of specular path perturbation. ACM Trans. Graph. **19**(4), 246–278 (2000)
3. Cline, D., Talbot, J., Egbert, P.: Energy redistribution path tracing. ACM Trans. Graph. **24**(3), 1186–1195 (2005)
4. Cook, R.L., Torrance, K.E.: A reflectance model for computer graphics. ACM Transactions on Graphics **1**(1), 7–24 (1982)
5. Doucet, A., Johansen, A., Tadic, V.: On solving integral equations using Markov Chain Monte Carlo methods. Applied Mathematics and Computation (2010)
6. Grünschloß, L., Raab, M., Keller, A.: Enumerating quasi-Monte Carlo point sequences in elementary intervals. In: L. Plaskota, H. Woźniakowski (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2010, *Springer Proceedings in Mathematics & Statistics*, vol. 23, pp. 399–408. Springer Berlin Heidelberg (2012)
7. Hachisuka, T., Kaplanyan, A.S., Dachsbacher, C.: Multiplexed Metropolis light transport. ACM Trans. Graph. **33**(4), 100:1–100:10 (2014)
8. Heckbert, P.S.: Adaptive radiosity textures for bidirectional ray tracing. In: Computer Graphics (Proceedings of SIGGRAPH 90) (1990)
9. Jakob, W.: Mitsuba renderer (2010). Http://www.mitsuba-renderer.org
10. Jakob, W.: Light transport on path-space manifolds. Ph.D. thesis, Cornell University (2013)
11. Jakob, W., Marschner, S.: Manifold exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. ACM Trans. Graph. **31**(4), 58:1–58:13 (2012)
12. Kajiya, J.T.: The rendering equation. In: Computer Graphics (Proceedings of SIGGRAPH 86), pp. 143–150 (1986)
13. Kaplanyan, A.S., Hanika, J., Dachsbacher, C.: The natural-constraint representation of the path space for efficient light transport simulation. ACM Transactions on Graphics (Proc. SIGGRAPH) **33**(4) (2014)
14. Kelemen, C., Szirmay-Kalos, L., Antal, G., Csonka, F.: A simple and robust mutation strategy for the Metropolis light transport algorithm. Computer Graphics Forum **21**(3), 531–540 (2002)
15. Keller, A.: Quasi-Monte Carlo Image Synthesis in a Nutshell. Springer (2014)
16. Kollig, T., Keller, A.: Efficient bidirectional path tracing by randomized quasi-Monte Carlo integration. Springer (2002)
17. Lafortune, E.P., Willems, Y.D.: Bi-directional path tracing. In: Proceedings of Compugraphics 93. Alvor, Portugal (1993)
18. Lehtinen, J., Karras, T., Laine, S., Aittala, M., Durand, F., Aila, T.: Gradient-domain Metropolis light transport. ACM Trans. Graph. **32**(4) (2013)
19. Manzi, M., Rousselle, F., Kettunen, M., Lethinen, J., Zwicker, M.: Improved sampling for gradient-domain Metropolis light transport. ACM Trans. Graph. (2014)
20. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)

21. Mitchell, D.P., Hanrahan, P.: Illumination from curved reflectors. In: Computer Graphics (Proceedings of SIGGRAPH 92), pp. 283–291 (1992)
22. Nicodemus, E.: Geometrical considerations and nomenclature for reflectance, vol. 160. US Department of Commerce, National Bureau of Standards Washington, D. C (1977)
23. Pauly, M., Kollig, T., Keller, A.: Metropolis light transport for participating media. In: Rendering Techniques 2000: 11th Eurographics Workshop on Rendering, pp. 11–22 (2000)
24. Pharr, M., Humphreys, G., Jakob, W.: Physically based rendering: From theory to implementation, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2016)
25. Preisendorfer, R.: Hydrologic optics. US Department of Commerce (1976)
26. Spivak, M.: Calculus on Manifolds. Addison-Wesley (1965)
27. Torrance, E., Sparrow, M.: Theory for off-specular reflection from roughened surfaces. JOSA **57**(9), 1105–1112 (1967)
28. Veach, E.: Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University (1997)
29. Veach, E., Guibas, L.: Bidirectional estimators for light transport. In: Fifth Eurographics Workshop on Rendering (1994)
30. Veach, E., Guibas, L.J.: Optimally combining sampling techniques for Monte Carlo rendering. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, SIGGRAPH '95, pp. 419–428. ACM (1995)
31. Veach, E., Guibas, L.J.: Metropolis light transport. In: Computer Graphics (Proceedings of SIGGRAPH 97), pp. 65–76 (1997)
32. Walter, B., Marschner, S.R., Li, H., Torrance, K.E.: Microfacet models for refraction through rough surfaces. In: Rendering Techniques 2007: 18th Eurographics Workshop on Rendering, pp. 195–206 (2007)
33. Walter, B., Zhao, S., Holzschuch, N., Bala, K.: Single scattering in refractive media with triangle mesh boundaries. ACM Trans. Graph. **28**(3) (2009)