# Guided Sampling and Caching for Efficient Physically Based Rendering

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik des
Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation

von

Vincent Schüßler

Tag der mündlichen Prüfung: 15. Juli 2026

1. Referent: Prof. Dr.-Ing. Carsten Dachsbacher
2. Referent: Assoc. Prof. Dr. Jeppe Revall Frisvad

# Acknowledgments

This dissertation would not have been possible without the people who accompanied me along the way.

First and foremost, my thanks go to my advisor, Carsten Dachsbacher, for giving me the opportunity to join his group and pursue this research, and for his guidance and support over the years. I also valued the trust he placed in me and the freedom he gave me to explore various research areas and ideas.

I am especially thankful to Johannes Hanika, who has guided me since my bachelor's thesis with ideas, encouragement, critical questions, and technical help. His many stories also taught me a great deal about rendering beyond academia.

I was fortunate to collaborate with a number of excellent co-authors. Alisa, aside from being an amazing office mate herself, helped me tremendously in wrapping up the vertex triplets project without losing my sanity. Addis and Sebastian kindly included me in their work and in many inspiring discussions about path guiding. It was a pleasure to supervise Mirco and Sergey in their respective student projects and to work with them on the subsequent publications. I thank Basile Sauvage and Jean-Michel Dischler for the fruitful collaboration on procedural texture graphs. Going further back, I remain grateful to Eric Heitz, who brought a great deal of insight and dedication to my first publication and helped to set me on this path.

I also thank Jeppe Revall Frisvad for agreeing to act as the second referee, and for the time and care he invested in reviewing this dissertation.

I thank everyone in the computer graphics group at KIT for the many good moments we shared. I have fond memories of our makeshift pasta lunches, reviving the office after the long COVID-19 break, many after-work bouldering sessions, the occasional beer, some attempts at making music, and countless coffee breaks. My thanks go in particular to Addis, Alisa, Bene, Christoph, Emanuel, Jo, Killian, Lorenzo, Lucas, Max, Nathan, Reiner, and Tobias. Many thanks also go to Diana for her kind and reliable help with the paperwork and for caring about the people behind it.

Beyond KIT, I thank everyone at Wētā FX for being so welcoming and for making my internship such an enjoyable experience. Special thanks go to Kerry, Louis-Daniel, Manuele, Max, Rémi, and Vincent, who put up with me and kept my spirits up in the weeks before my defense.

I am deeply grateful to all my friends, who always managed to take my mind off work when I needed it. You know who you are. Above all, I am truly thankful to my parents, who have always been there for me and supported me in everything I set out to do.

# List of Publications

This dissertation is partially based on the following publications.

Schüßler, Vincent, Hanika, Johannes, Sauvage, Basile, Dischler, Jean-Michel, and Dachsbacher, Carsten. "Selective Caching in Procedural Texture Graphs for Path Tracing". *Proceedings of the Eurographics Symposium on Rendering (EGSR)*. 2025. DOI: 10.2312/sr.20251176.

Lelyakin, Sergey, Schüßler, Vincent, and Dachsbacher, Carsten. "Sampling of Anisotropic Spatial Gaussians for Path Guiding". *Eurographics - Posters*. 2025. DOI: 10.2312/egp.20251017.

Werner, Mirco, Schüßler, Vincent, and Dachsbacher, Carsten. "ReSTIR Subsurface Scattering for Real-Time Path Tracing". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 7.3 (2024). DOI: 10.1145/3675372.

Schüßler, Vincent, Hanika, Johannes, and Dachsbacher, Carsten. "Bridge Sampling for Connections via Multiple Scattering Events". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 43.4 (2024). DOI: 10.1111/cgf.15160.

Dittebrandt, Addis, Schüßler, Vincent, Hanika, Johannes, Herholz, Sebastian, and Dachsbacher, Carsten. "Markov Chain Mixture Models for Real-Time Direct Illumination". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 42.4 (2023). DOI: 10.1111/cgf.14881.

Schüßler, Vincent, Hanika, Johannes, Jung, Alisa, and Dachsbacher, Carsten. "Path Guiding with Vertex Triplet Distributions". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 41.4 (2022). DOI: 10.1111/cgf.14582.

# Contents

**Acknowledgments** i

**List of Publications** iii

**1 Introduction** 1
- 1.1 Summary of contributions . . . 2
  - 1.1.1 Vertex triplet distributions for path guiding . . . 2
  - 1.1.2 Multiple-scattering connections in participating media . . . 2
  - 1.1.3 Dependent sampling with stochastic multiple importance sampling . . . 3
  - 1.1.4 Invertible sampling with ReSTIR . . . 3
  - 1.1.5 Optimized cache placement in procedural texture graphs . . . 3
- 1.2 Organization of the dissertation . . . 4

**2 Background** 5
- 2.1 Light transport model . . . 5
  - 2.1.1 Path space . . . 6
  - 2.1.2 Integration and measure theory . . . 7
  - 2.1.3 Geometric measures . . . 8
  - 2.1.4 Radiometry . . . 10
  - 2.1.5 Scene model and photon interactions . . . 12
  - 2.1.6 Measurement contribution function . . . 13
- 2.2 Monte Carlo integration . . . 14
  - 2.2.1 Random variables and expectation . . . 14
  - 2.2.2 Monte Carlo estimator . . . 16
  - 2.2.3 Importance sampling . . . 17
- 2.3 Transport path sampling . . . 19
  - 2.3.1 Path sampling operations . . . 19
  - 2.3.2 Path sampling algorithms . . . 21
  - 2.3.3 Challenges . . . 22
  - 2.3.4 Advanced path sampling . . . 23
- 2.A Derivation of solid angle density . . . 26

**3 Path Guiding with Vertex Triplet Distributions** 27
- 3.1 Introduction . . . 27
- 3.2 Related work . . . 28
  - 3.2.1 Path guiding . . . 28
  - 3.2.2 Glossy light transport . . . 30

3.3 Overview and theoretical background . . . 30
3.3.1 Target function . . . 31
3.3.2 Gaussian mixture model for path guiding . . . 33
3.3.3 Algorithm . . . 33
3.4 The guiding distribution . . . 34
3.4.1 Light transport modes . . . 34
3.4.2 Mixture at the camera . . . 35
3.4.3 Truncated Gaussians . . . 36
3.5 Path sampling with vertex triplet distributions . . . 36
3.5.1 Sampling the conditional GMM . . . 36
3.5.2 Sampling a surface vertex position . . . 37
3.5.3 Handling (near) specular vertices . . . 37
3.5.4 Image plane sampling . . . 38
3.6 Learning a vertex triplet distribution with expectation maximization . . . 38
3.6.1 Collecting triplet samples . . . 38
3.6.2 Expectation maximization . . . 40
3.6.3 Initial mixture . . . 42
3.7 Results . . . 43
3.7.1 Tested methods . . . 43
3.7.2 Comparisons . . . 44
3.8 Limitations and future work . . . 48
3.9 Conclusion . . . 48
3.A Conditional and marginal normal distributions . . . 49
3.B Exact sampling of 3D Gaussians for surface vertices . . . 49

**4 Bridge Sampling for Connections via Multiple Scattering Events 51**
4.1 Introduction . . . 52
4.2 Related work . . . 52
4.3 Sampling vertex bridges . . . 54
4.3.1 PDF derivation . . . 56
4.3.2 Distance sampling based on scaling . . . 57
4.3.3 Constraint-based distance sampling for $n = 2$ and $n = 3$ . . . 58
4.4 Sampling the number of vertices . . . 60
4.4.1 Defining the bridge estimator . . . 61
4.4.2 Precomputing $\sqrt{\mathbb{E}[B_n^2]}$ for homogeneous media . . . 61
4.4.3 Importance sampling $n$ . . . 63
4.5 Results . . . 63
4.5.1 Parametric unit test case . . . 63
4.5.2 Renderer integration and heterogeneous media . . . 67
4.6 Conclusion and future work . . . 70
4.A Alternative view on scaling-based sampling . . . 71
4.B Equivalence to OMNEE . . . 71

4.C Comparison to the track rotation estimator . . . 71
4.C.1 Modified track rotation estimator . . . 72
4.C.2 Extension of the TRE with bridge sampling . . . 73
4.C.3 Results . . . 74
4.C.4 Conclusion . . . 75

**5 Interactive Light Transport Simulation 77**
5.1 Markov chain mixture models for real-time direct illumination . . . 77
5.1.1 Stochastic multiple importance sampling with dependent samples . . . 78
5.1.2 Summary . . . 81
5.2 ReSTIR subsurface scattering for real-time path tracing . . . 81
5.2.1 Combining shift mappings . . . 82
5.2.2 Path space extension . . . 83
5.2.3 Probabilistic inversion in ReSTIR . . . 85
5.2.4 Summary . . . 85
5.3 Conclusion . . . 85

**6 Selective Caching in Procedural Texture Graphs for Path Tracing 87**
6.1 Introduction . . . 88
6.2 Background and related work . . . 88
6.3 Overview . . . 90
6.4 Texture graph evaluation with filtering . . . 91
6.4.1 Filtering theory . . . 92
6.4.2 Filtering approximation . . . 93
6.5 Texture cache . . . 94
6.5.1 Footprint evaluation strategies . . . 94
6.5.2 Choosing a strategy . . . 96
6.6 Measure for computational cost . . . 96
6.6.1 Cost components and coefficients . . . 97
6.6.2 Aggregate cost . . . 97
6.6.3 Computing the value per node in practice . . . 98
6.7 Prediction of active texture area . . . 99
6.7.1 Idea . . . 99
6.7.2 Zero-truncated Poisson mixture model . . . 100
6.7.3 Model parameter estimation . . . 100
6.7.4 Prediction . . . 101
6.8 Optimization of caching decision . . . 103
6.8.1 Integer linear programming formulation . . . 103
6.9 Evaluation . . . 104
6.10 Conclusion . . . 109
6.A Footprint derivation . . . 110
6.A.1 Introduction . . . 110
6.A.2 Model . . . 110

6.A.3 Derivation . . . . . 112

**7 Conclusion** **115**

**Bibliography** **117**

# 1 Introduction

Physically based rendering is the process of creating synthetic images by simulating light transport from sources to sensors in a virtual scene. Its goal is to achieve photorealistic results in the form of images that are indistinguishable from real photographs. For example, film production demands seamless blending of synthetic images with real photography while retaining full creative control. Machine learning makes use of rendering to create or augment training data sets with synthetic images. In architectural or product visualization, realistic renderings of virtual prototypes support the design process, extending even to the assessment of optical properties.

Meeting these requirements involves simulating the transport of light within a detailed scene model. The scene comprises the 3D geometry as well as material models, which use input parameters stored in high-resolution textures. Typically, we use geometrical optics as the underlying physical model of light transport. Light travels along straight rays and can be scattered on surfaces or in free space inside of participating media. A sequence of scattering interactions that links a light source to a sensor defines a light transport path. The light transport problem can be formulated as an integral of a measurement contribution function over the set of all paths, referred to as path space.

Monte Carlo integration has emerged as the dominant solution method, in particular due to its flexibility with regard to both material models and geometric complexity and representation. It defines an estimator based on the contribution of stochastically sampled paths. This estimator is equal to the value of the integral in expectation. In practice we can only afford to sample a limited number of paths. The residual error manifests in a noisy image and corresponds to estimator variance. When noise levels are moderate, denoising algorithms can recover smooth images that closely approximate the ground truth. Yet even in seemingly simple scenes, the variance of path contributions can become prohibitively high, resulting in substantial error in practice. The primary objective of efficient Monte Carlo-based rendering algorithms is therefore twofold: keeping the computational cost of path sampling and evaluation low, while simultaneously maintaining robustness by keeping the variance within acceptable limits. Achieving both is vital for practical convergence, but remains challenging.

Importance sampling is commonly used to address the most severe sources of variance. It seeks to sample paths with higher contributions with proportionally higher probability densities. Perfect importance sampling would yield an estimator with zero variance, but is unfortunately at least as hard as computing the value of the original integral. Multiple sampling techniques are instead combined into a more robust estimator, with each technique targeting only part of the path contribution. However, this approach suffers from two limitations. First, for some paths no technique exists that samples them reliably. Second, combining many techniques at once can incur significant computational cost, especially in scenes where simple techniques would suffice.

These limitations stem from the fact that many classic rendering algorithms are oblivious to the global distribution of illumination in the scene. This is unsurprising, as we require simulation to

obtain this unknown distribution. However, selecting an efficient combination of importance sampling techniques would require prior knowledge of paths encountered during sampling. Consequently, a significant portion of computation may be spent on less effective techniques simply to preserve robustness. Moreover, relying solely on local information makes many classes of paths inherently difficult to sample. Robust sampling of such paths requires more advanced techniques that incorporate global information.

In this dissertation, we focus on data-driven approaches that guide rendering algorithms using information derived from previously sampled paths. A substantial amount of data is available for this purpose, as many related integrals are estimated in parallel to compute an image, and multiple samples are required for convergence. This data can be leveraged to adapt the algorithm to the specific scene, improving its performance for future paths. In this way, many of the advantageous properties of Monte Carlo-based algorithms are preserved, while inefficiencies arising from their reliance on brute-force independent sampling are mitigated. When these ideas are applied to importance sampling to learn better sampling distributions, the approach is known as path guiding. We introduce a path guiding method that learns and represents distributions of paths in a general yet precise manner, reducing variance in complicated light transport scenarios. Beyond variance reduction, data-driven approaches can also be used to dynamically optimize other algorithmic parameters. Monte Carlo-based algorithms exhibit challenging memory access patterns due to their random nature. We address this by optimizing the placement of texture-data caches using observations of the sampled path distribution, thereby lowering the cost of path evaluation.

## 1.1 Summary of contributions

We summarize the contributions of this dissertation in the following.

### 1.1.1 Vertex triplet distributions for path guiding

A key challenge of path guiding is the efficient and accurate reconstruction of a high-dimensional path distribution for guided sampling. While existing approaches typically model guiding as a product of low-dimensional distributions, these fail to capture complex transport effects that induce dependencies between path vertices. However, direct modeling in path space does not scale due to its high dimensionality. Based on the structure of the measurement contribution function, we show that only dependencies between three consecutive vertices are relevant for modeling light transport. We therefore introduce a general path guiding model based on the joint distribution of triplets of vertices. It supports both surface and volumetric transport, as well as bidirectional guided path construction. We further propose an effective training method for this model using reservoir-based resampling and a clustering approach.

### 1.1.2 Multiple-scattering connections in participating media

The information of light sources in a scene is usually leveraged to locally sample the direct illumination. However, this becomes less effective in dense, forward-scattering media, where the indirect illumination can dominate. We introduce a method to explicitly construct corresponding paths involving multiple

scattering events. To achieve low variance, importance sampling of the scattering directions is crucial. Previously, no efficient solution was known for constructing such connecting paths with prescribed endpoints and directions. Our algorithm can sample connecting paths with an arbitrary number of vertices and an analytic probability density function. We also provide an algorithm to sample the number of vertices based on connection distance and medium properties.

### 1.1.3 Dependent sampling with stochastic multiple importance sampling

While mixture models offer high expressivity for path guiding, the optimization of their parameters remains computationally demanding. Especially in interactive applications, it can be appealing to represent mixture models implicitly instead, for example as the stationary distribution of a stochastic process. The stochastic multiple importance sampling estimator can be used to combine such an uncountable set of sampling techniques. However, its original formulation requires techniques and samples to be stochastically independent. We identify conditions under which this independence requirement can be relaxed, and apply the result to path guiding with implicit mixture models.

### 1.1.4 Invertible sampling with ReSTIR

Reservoir-based spatio-temporal importance resampling (ReSTIR) enables the reuse of paths across pixels and frames and is particularly suited for interactive applications. Reuse is achieved through shift mappings, which transfer paths between domains. The effectiveness of a particular mapping depends on the specific path. However, the combination of mappings is not feasible when they are defined in spaces of different dimensionality. Similar issues have been encountered in other areas of rendering, where an ambiguous mapping from path space to the space of random numbers was resolved using probabilistic inversion. We show that this approach is also applicable to ReSTIR, even though shift mappings are required to be deterministic.

### 1.1.5 Optimized cache placement in procedural texture graphs

In addition to reducing variance, low-overhead path evaluation is essential for convergence within acceptable computational time. Materials often use implicit representations in the form of procedural texture graphs to model spatial variation. While offering a compact representation, they incur significant evaluation cost. Fully precomputing these textures is usually impractical due to the high memory requirements. Instead, texture graphs are evaluated on demand at path vertices during path construction. Because texture resolution is finite, identical texture values may be recomputed repeatedly in this process. Caching can eliminate these redundancies, but its effectiveness depends on amortization through reuse at sufficiently high sampling densities. This in turn depends on graph topology, placement within the scene, and overall cache capacity.

We introduce a cost-based framework for optimizing caching of texture graph nodes. We define a cost function that primarily balances the total number of texture accesses and the number of cache entries required to support them. To estimate these costs, we develop a statistical model that predicts the number of cache entries required during rendering, based on texture accesses recorded in a profiling

phase. Using integer linear programming, we then minimize the cost function to identify an optimal set of nodes for caching.

## 1.2 Organization of the dissertation

The discussion of the contributions in this dissertation is structured into chapters as outlined below.

**Chapter 2** reviews the fundamental concepts and theoretical background required for understanding the state of the art in Monte Carlo-based rendering.

**Chapter 3** introduces a method for path guiding based on modeling the joint distribution of triplets of consecutive path vertices. A version of this chapter has been previously published as [SHJD22b]. Appendix 3.B is based on the publication [LSD25].

**Chapter 4** presents a method for sampling connecting paths with an arbitrary number of vertices in dense, forward-scattering participating media. A version of this chapter has been previously published as [SHD24a]. Appendix 4.C contains a previously unpublished comparison to the track-rotation estimator from the neutron transport literature.

**Chapter 5** discusses contributions that are motivated by their use in interactive applications. This encompasses implicit mixture models for path guiding, as well as invertible shift mappings for ReSTIR. It is based on contributions to previous publications [DSH*23; WSD24].

**Chapter 6** proposes a cost-based framework for optimizing the selection of nodes to cache in a procedural texture graph. A version of this chapter has been previously published as [SHS*25b].

**Chapter 7** summarizes and concludes this dissertation.

# 2 Background

The goal of physically based rendering is to produce photorealistic images as the result of simulating light transport inside a virtual scene. In this chapter, we review the respective theoretical background and foundational algorithms for this dissertation. To specify the measurements that we want to simulate, we introduce definitions of relevant quantities and units. We model the physical light transport as stochastic particles ("photons") interacting with the scene. This allows us to express the unknown measurements as the result of integrating the contribution of all *light transport paths*, i.e. trajectories of photons. The light transport simulation corresponds to solving this integral. Due to its high dimensionality, we use Monte Carlo integration and sample paths stochastically.

Since both the light transport model and its simulation are stochastic, one possibility is to sample paths in analogy to the model. However, for efficiency we must focus sampling effort on the small subset of paths that contribute meaningfully to the virtual measurement. This is challenging to achieve for general light transport. Convergence of the integration is determined both by the representativeness of sampled paths for the physical transport and by the computational overhead of path sampling. Therefore, path sampling algorithms are at the core of efficient light transport simulation. In the remainder of this chapter, we review the state of the art in path sampling.

## 2.1 Light transport model

The virtual scene is a set of *media* in 3D space with homogeneous index of refraction (IOR), and a set of 2D manifold *surfaces*, which represent interfaces between media. At every point $\mathbf{x}$ on a surface, its orientation is defined by a normal vector $n(\mathbf{x})$ of unit length. Computer graphics commonly uses a simplified version of the geometrical optics framework. It is characterized by describing light as

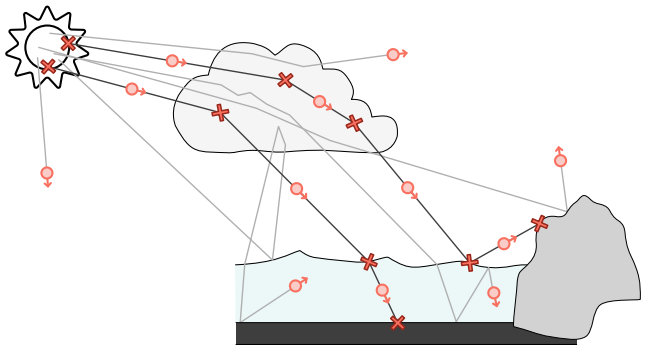


**Figure 2.1:** *Events describe the interaction of photons with the scene. They occur either on surfaces or in media. After a photon is emitted from a light source, it travels in a straight line until it is either absorbed, or scattered into a different direction. This sequence of events defines a light transport path.*

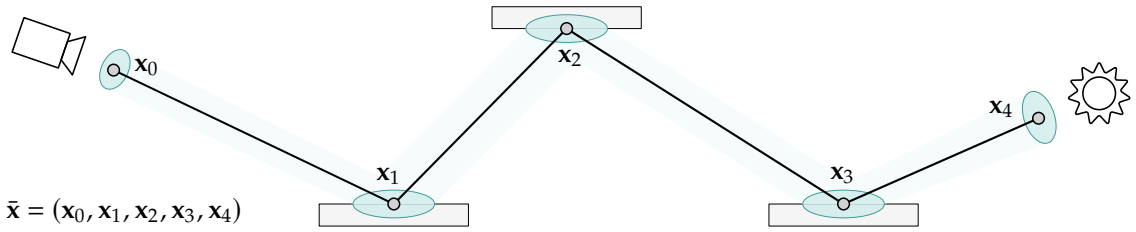



**Figure 2.2:** *We integrate a contribution function $f(\bar{\mathbf{x}})$ over all light transport paths, each one linking camera and light source via a sequence of scattering events. The contribution is expressed as a density in relation to a measure on the path space. We can intuitively imagine this as measuring the transported power in a small neighborhood around the path, defined by the product of vertex neighborhoods.*

particles ("photons") that travel through space in straight lines. They are not affected by external forces and only interact with the scene geometry via collision. Therefore, the complete trajectory of a photon can be described as a sequence of interaction *events*: emission, scattering and absorption (see Fig. 2.1). When events occur inside media in free space, we refer to this as *volume* transport and the medium as *participating*. The sequence of events is a light transport path. The goal of this section is to define the simulated measurement in terms of paths, before we proceed to algorithms for computing it.

### 2.1.1 Path space

We use the path integral formulation of light transport [Vea97; PKK00] in this dissertation. This allows us to describe and reason about many different light transport algorithms in the same mathematical framework. We first define *path space* as the set of all light transport paths and then express the *measurement equation* in terms of an integral over path space. See Fig. 2.2 for an illustration.

**Path space.** Each path consists of multiple path *vertices*, where each vertex corresponds to an event. Formally, we define the path space $\mathcal{P}$ as the union of all sets $\mathcal{P}_k$, which contain the paths of length $k+1$:

$$\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}_k = \bigcup_{k=1}^{\infty} \{\bar{\mathbf{x}} = (\mathbf{x}_0, \ldots, \mathbf{x}_k) \in \mathcal{P}_k\} \,. \tag{2.1}$$

We denote path vertices by $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, and directions by $\boldsymbol{\omega}$. $\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}$ is the normalized direction from $\mathbf{x}$ to $\mathbf{y}$.

**Measurement equation.** For each pixel $j$ of our sensor, the *measurement contribution* $f_j$ is a function of a path $\bar{\mathbf{x}}$. The total measurement $I_j$ of this pixel is then the integral of the contribution over the whole path space, as summarized in the measurement equation:

$$I_j = \int_{\mathcal{P}} f_j(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}. \tag{2.2}$$

The goal of light transport simulation is to compute the value of $I_j$ for each pixel. In the following sections, we specify our model of light transport more precisely, after introducing required mathematical background. The result is the definition of the measurement contribution function in Eq. (2.25).

### 2.1.2 Integration and measure theory

The concise formulation of the measurement equation (Eq. (2.2)) is possible because we use the *Lebesgue* integral. This allows us to define integration with respect to a *measure* on the abstract path space. Measure theory is also the foundation of modern probability theory and therefore provides the link to Monte Carlo integration, which we use to compute estimates of the integral. We summarize key definitions and theorems without proof in this section, since we only require a high-level understanding of fundamental concepts. Please refer to the book by Billingsley [Bil95] for a comprehensive and rigorous introduction to measure theory with a focus on probability.

**Measure.** A measure is a mathematical concept to describe the "size" of sets in a general and consistent way. For a more formal definition, let $\sigma(\Omega)$ be a $\sigma$-algebra[1] over a set $\Omega$. A measure is a function $\mu\colon \sigma(\Omega) \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ that assigns a value of zero to the empty set and is countably additive:

**i.** $\mu(\emptyset) = 0$,

**ii.** $\mu\left(\bigcup_{k=1}^{\infty} A_k\right) = \sum_{k=1}^{\infty} \mu(A_k)$ for every disjoint sequence $A_1, A_2, \ldots$ in $\sigma(\Omega)$.

**Example: Lebesgue measure.** The $k$-dimensional Lebesgue measure $\lambda_k$ on $\mathbb{R}^k$ corresponds to our intuitive notion of length ($k = 1$), area ($k = 2$) or volume ($k \geq 3$). The intuitive definition on $k$-rectangles

$$\lambda_k(\{x \mid a_i < x_i \leq b_i, i = 1, \ldots, k\}) = \prod_{i=1}^{k} (b_i - a_i), \tag{2.3}$$

uniquely determines $\lambda_k(A)$ for all $A \in \sigma(\mathbb{R}^k)$.

**Lebesgue integral.** To integrate a real, nonnegative[2] function $f$ with respect to a measure $\mu$, we decompose the domain into a finite number of disjoint subsets $A_i \in \sigma(\Omega)$. On each subset $A_i$, we approximate $f$ by $\inf_{\omega \in A_i} f(\omega)$. Then we sum its product with the subset size with respect to the measure $\mu(A_i)$ over all $A_i$. The Lebesgue integral is defined as the supremum of this sum over all such finite decompositions:

$$\int_{\Omega} f \,\mathrm{d}\mu = \sup \sum_i \left[\inf_{\omega \in A_i} f(\omega)\right] \mu(A_i). \tag{2.4}$$

The Lebesgue measure links this definition to the Riemann integral. If $f$ is both Lebesgue and Riemann integrable, the Lebesgue integral with respect to Lebesgue measure has the same value as the Riemann integral. We commonly use alternative notation of the integral to make the function arguments explicit, or omit the measure for brevity when it is the Lebesgue measure, or unambiguous from context:

$$\int_{\Omega} f \,\mathrm{d}\mu = \int_{\Omega} f(x) \,\mathrm{d}\mu(x) = \int_{\Omega} f(x) \,\mu(\mathrm{d}x) = \int_{\Omega} f(x) \,\mathrm{d}x. \tag{2.5}$$

[1] Billingsley [Bil95] uses the term $\sigma$-field in place of $\sigma$-algebra and provides a formal definition. For our purposes, it is sufficient to think of it as a subset of the power set $\mathcal{P}(\Omega)$ that excludes "unusual" subsets, which would lead to contradictions. While other choices are possible in general, we assume the *Borel* $\sigma$-algebra $\sigma(\Omega)$ for simplicity, which is the smallest $\sigma$-algebra over $\Omega$.

[2] With minor technical changes, $f$ is not required to be nonnegative. See the literature for the more general definition.

**Measurable functions.** Measurability is a useful property that allows us to define measures via a function. A function $T\colon \Omega \to \Omega'$ is *measurable* if for each $A' \in \sigma(\Omega')$ its inverse image $T^{-1}(A')$ is in $\sigma(\Omega)$. Then, using $T^{-1}$ we can *push forward* the measure $\mu$ on $\Omega$ to define a new measure on $\Omega'$:

$$\mu \circ T^{-1}(A') = \mu(T^{-1}(A')), \quad A' \in \sigma(\Omega'). \tag{2.6}$$

$f$ is integrable with respect to $\mu \circ T^{-1}$ iff $f \circ T$ is integrable with respect to $\mu$. In this case, we can perform a change of variables:

$$\int_{T^{-1}(A')} f(T(\omega))\,\mathrm{d}\mu(\omega) = \int_{A'} f(\omega')\,\mathrm{d}\mu \circ T^{-1}(\omega'). \tag{2.7}$$

**Densities.** A nonnegative measurable function $\rho$ is a *density* of $\nu$ with respect to $\mu$ if

$$\nu(A) = \int_A \rho(\omega)\,\mathrm{d}\mu(\omega), \quad A \in \sigma(\Omega), \tag{2.8}$$

defines the measure $\nu$. $f$ is integrable with respect to $\nu$ iff $f\rho$ is integrable with respect to $\mu$. In this case, we can use the density to change the measure of integration:

$$\int_A f(\omega)\,\mathrm{d}\nu(\omega) = \int_A f(\omega)\,\rho(\omega)\,\mathrm{d}\mu(\omega). \tag{2.9}$$

**Radon-Nikodym derivative.** Equation (2.9) shows that densities can be useful to change measure. The Radon-Nikodym theorem states sufficient conditions for existence and uniqueness of such a density. Suppose $\nu$ is another measure. If $\mu(A) = 0$ implies $\nu(A) = 0$ for every $A \in \sigma(\Omega)$, we say that $\nu$ is *absolutely continuous* with respect to $\mu$, denoted by $\nu \ll \mu$. In this case, there exists a density that satisfies Eq. (2.8) and is unique up to sets of measure zero. $\rho$ is called the *Radon-Nikodym derivative* of $\nu$ with respect to $\mu$ and can be written as $\frac{\mathrm{d}\nu}{\mathrm{d}\mu}$.

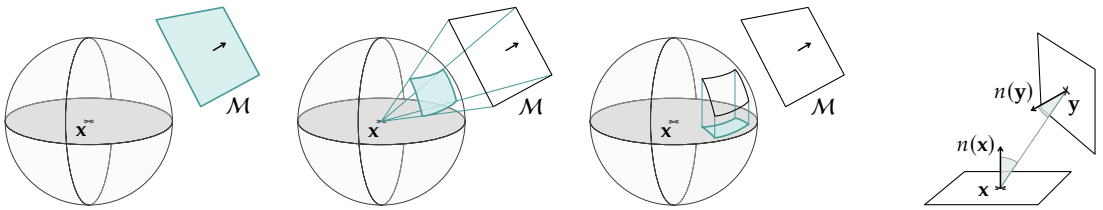
**Change of variables with Lebesgue measure.** When integrating in $\mathbb{R}^k$ with respect to Lebesgue measure, we can make the change of variables from Eq. (2.7) more specific. Let $U \subset \mathbb{R}^k$ be an open set and $T\colon U \to \mathbb{R}^k$ be continuously differentiable and injective. Suppose that $\det J_T(x) \neq 0$ for all $x$, where $J_T$ is the Jacobian matrix of partial derivatives of $T$. If $f$ is nonnegative, then

$$\int_U f(T(x))\,|\det J_T(x)|\,\mathrm{d}x = \int_{T(U)} f(y)\,\mathrm{d}y. \tag{2.10}$$

We see that in this case, the density of the push-forward measure (of $T^{-1}$) is the Jacobian determinant.

### 2.1.3 Geometric measures

In the following, we define measures that are useful for describing the geometry of light transport paths. We illustrate these concepts in Fig. 2.3.




**(a)** *Area* **(b)** *Solid angle* **(c)** *Projected solid angle* **(d)** *Geometry term*

**Figure 2.3:** *We measure a surface $\mathcal{M}$ either using its area (a), its subtended solid angle from a point $\mathbf{x}$ (b), or its solid angle projected on a surface (c). The symmetric geometry term $G(\mathbf{x}, \mathbf{y})$ relates a density expressed with respect to area measure of $\mathbf{y}$ to the projected solid angle density from $\mathbf{x}$, or vice versa (d).*

**Area measure.** The area measure $A$ on a 2D surface $\mathcal{M} \subset \mathbb{R}^3$ is equivalent to the usual notion of surface area. It can be defined using a parametrization $\varphi\colon \mathbb{R}^2 \to \mathcal{M},\ (u, v) \mapsto (x, y, z)$ [MT12, Sec. 7.4]:

$$A(M) = \int_{\varphi^{-1}(M)} \left\| \frac{\partial \varphi}{\partial u} \times \frac{\partial \varphi}{\partial v} \right\| \mathrm{d}(u, v). \tag{2.11}$$

**Solid angle measure.** The *solid angle* extends the concept of angles to higher dimensions and uses the unit steradian [sr]. We use the solid angle measure $\sigma$ to measure sets of directions. It is defined as the area measure on the unit sphere $S^2$. We are often interested in the solid angle *subtended* by a surface, which is the solid angle of the projection onto the unit sphere. In Appendix 2.A, we show that the surface area measure of $\mathbf{y}$ and its solid angle measure at $\mathbf{x}$, where $n$ is the surface normal, are related by

$$\frac{\mathrm{d}\sigma_{\mathbf{x}}(\mathbf{y})}{\mathrm{d}A(\mathbf{y})} = \frac{\left| n(\mathbf{y}) \cdot \boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}} \right|}{\|\mathbf{x} - \mathbf{y}\|^2}. \tag{2.12}$$

**Projected solid angle measure.** When we measure solid angle from a receiving surface, it can be important to additionally take its orientation into account. For example, photons arriving from a shallow angle will spread over a wider area than if arriving orthogonally to the surface, affecting their density on the surface. The *projected solid angle* measure $\sigma^{\perp}_{\mathbf{x}}$ at $\mathbf{x}$ incorporates this weighting into the measure:

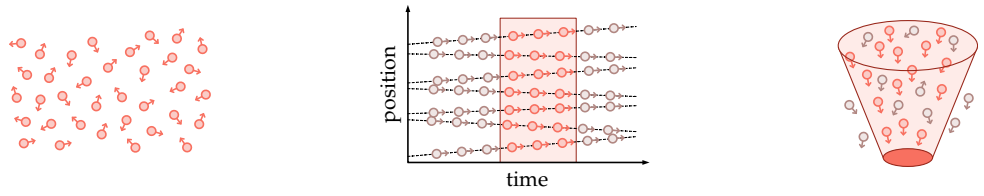
$$\frac{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\mathbf{y})}{\mathrm{d}\sigma_{\mathbf{x}}(\mathbf{y})} = \left| n(\mathbf{x}) \cdot \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} \right|. \tag{2.13}$$

**Geometry term.** To convert between measures in the directional and spatial domain for both surfaces and media in a general way, we define the *geometry term* $G(\mathbf{x}, \mathbf{y})$:

$$G(\mathbf{x}, \mathbf{y}) = \frac{D_{\mathbf{x}}(\mathbf{y})\, D_{\mathbf{y}}(\mathbf{x})}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{2.14}$$

$$D_{\mathbf{x}}(\mathbf{y}) = \begin{cases} \left| n(\mathbf{x}) \cdot \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}} \right| & \text{if } \mathbf{x} \text{ is on a surface,} \\ 1 & \text{if } \mathbf{x} \text{ is in a medium,} \end{cases} \tag{2.15}$$

If $\mathbf{y}$ is on a surface, it relates area to projected solid angle measure, i.e. $G(\mathbf{x}, \mathbf{y}) = \frac{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\mathbf{y})}{\mathrm{d}A(\mathbf{y})}$. Here, we assumed $\sigma^{\perp}_{\mathbf{x}} = \sigma_{\mathbf{x}}$ if $\mathbf{x}$ is in a medium, since the surface normal is not defined in media, but can be




**(a)** *Phase space* **(b)** *Trajectory space and flux* **(c)** *Radiance*

**Figure 2.4:** *The complete system state is defined by the distribution of photons in phase space (a). Trajectory space extends it with a time dimension (b). We define radiant flux by measuring the photon density with respect to the temporal domain. Radiance measures photon density with respect to both projected solid angle and surface area (c).*

thought of as coinciding with $\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}$. When $\mathbf{y}$ is in a medium, $G$ relates the volume measure $V = \lambda_3$ to the product measure of projected solid angle and distance: $G(\mathbf{x},\mathbf{y}) = \frac{\mathrm{d}\sigma^{\perp}_{\mathbf{x}}\times\lambda(\mathbf{y},\|\mathbf{x}-\mathbf{y}\|)}{\mathrm{d}V(\mathbf{y})}$.

**Product measures.** As seen in the definition of the geometry term, it can be useful to form products of measures. In particular, this allows us to define a measure on the path space. Let $\mu_X$ and $\mu_Y$ be finite measures, defined on $\sigma(X)$ and $\sigma(Y)$ respectively. Their product $\mu_X \times \mu_Y$ defines a unique and finite measure on the product space $\sigma(X \times Y)$, where $\mu_X \times \mu_Y(A \times B) = \mu_X(A)\,\mu_Y(B)$ for $A \in \sigma(X), B \in \sigma(Y)$.

**Path space measure.** We have not yet defined a measure to use in the measurement equation (Eq. (2.2)), which requires a measure that is defined on path space $\mathcal{P}$. We choose either area measure $A$ or volume measure $V$ for each vertex and define a product measure $\mu_k$ on $\mathcal{P}_k$ by

$$\mu_k = \nu_0 \times \cdots \times \nu_k, \tag{2.16}$$

$$\nu_i = \begin{cases} A & \text{if } \mathbf{x}_i \text{ is on a surface,} \\ V & \text{if } \mathbf{x}_i \text{ is in a medium.} \end{cases} \tag{2.17}$$

To extend this definition to paths of all lengths, we sum the measures of each $\mathcal{P}_k$:

$$\mu(P) = \sum_{k=1}^{\infty} \mu_k\left(P \cap \mathcal{P}_k\right), \quad P \in \sigma(\mathcal{P}). \tag{2.18}$$

This is the path space measure that we use in the measurement equation.

### 2.1.4 Radiometry

So far, we have only discussed geometric considerations. In addition, we require some concepts from the field of *radiometry* to formalize quantities of light, and to relate these to simulated photon trajectories. Here, we review only the most important definitions that are relevant to this dissertation. For a comprehensive discussion of the subject in the context of graphics, we refer to the work by Arvo [Arv95b, Ch. 2] and Veach [Vea97, Ch. 3]. See Fig. 2.4 for an illustration of the most important concepts.

**Phase and trajectory space.** The state of a single photon is defined by its position $\mathbf{x}$, direction $\boldsymbol{\omega}$ and wavelength $\lambda$, which corresponds to a point $(\mathbf{x}, \boldsymbol{\omega}, \lambda) \in \mathbb{R}^3 \times S^2 \times \mathbb{R}^+$ in the *phase space*. The complete

system state can be defined by the distribution of all photons in phase space. We assume photons to be sufficiently numerous, such that we can model this distribution as a continuous one. *Trajectory space* extends phase space by a time dimension in order to describe photon movement.

**Event space and radiant energy.** For each radiometric quantity of interest, we define a region of trajectory space, an *event space*, in which we can measure *spectral*[3] *radiant energy* $Q_\lambda$. Intuitively, we imagine the event space as a photon detector. Quantities are derived by relating the energy measure to a suitable geometric measure on the event space, resulting in a density.

**Radiant flux.** Since the scale of our scene is typically small in comparison to the speed of light, we simulate the equilibrium state of the system, seeking the energy distribution in any instant of time. By defining the event space as a slice of time, we obtain *radiant power* or *radiant flux*:

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t} \left[\mathrm{W} = \mathrm{J\,s^{-1}}\right]. \tag{2.19}$$

**Radiance.** We usually require the photon density with respect to position and direction. To this end, we define an imaginary surface with a cone of directions attached as an event space, i.e. we measure the energy of photons arriving at the surface within the cone. We obtain *radiance* as the density of energy with respect to the product of surface area and projected solid angle measure:

$$L(\mathbf{x}, \boldsymbol{\omega}) = \frac{\mathrm{d}\Phi(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}A \times \sigma^{\perp}_{\mathbf{x}}(\mathbf{x}, \boldsymbol{\omega})} \left[\mathrm{W\,m^{-2}\,sr^{-1}}\right] \tag{2.20}$$

Radiance is conserved between two points in vacuum, i.e. in the absence of any interactions in-between. This makes it a convenient quantity for expressing the equations of transport, and for propagating it in our simulation.

**Color.** Color is a key aspect of visual perception, arising from the spectral distribution of light. We allow parameters of the scene, e.g. surface albedo, to depend on wavelength. For the simulation, we associate each transport path with a wavelength from the visible spectrum. These can be converted to a tristimulus value using the color matching functions of the CIE standard observer [SG31]. Tristimulus values represent the visual sensation of a certain color that can be efficiently stored and reproduced on an RGB display. Sometimes light transport is also directly simulated using tristimulus values, although this is less physically accurate. However, this approach effectively simulates many wavelengths with just a single path, which can be more efficient, since photon trajectories often do not depend strongly on wavelength. We use hero wavelength sampling [WND*14] to achieve similar gains in efficiency with full spectral simulation. Since the wavelength dependence is largely not important for the methods and discussion in this dissertation, we usually do not specify it explicitly in equations.

[3] For brevity, we omit the wavelength dimension below and derive quantities based on the non-spectral radiant energy $Q$. It can simply be replaced by $Q_\lambda$ to arrive at the spectral counterpart of each quantity.

### 2.1.5 Scene model and photon interactions

We are now ready to describe the model of photon interactions with the scene, i.e. emission, scattering and absorption on surfaces and in media. Alongside the discussion of possible interactions, we introduce relevant equations, before assembling the measurement contribution function from these terms.

**Participating media.** Media consist of tiny particles that photons can collide with. We describe particles statistically using a spatial density $\rho(\mathbf{x})$ $\left[\mathrm{m}^{-3}\right]$, instead of modeling them individually. We assume independent distribution and isotropic mean cross-sectional area $\sigma$ $\left[\mathrm{m}^{2}\right]$, i.e. the projected area of the particle is invariant under rotation. This allows us to derive a collision coefficient $\mu(\mathbf{x}) = \rho(\mathbf{x})\,\sigma$ $\left[\mathrm{m}^{-1}\right]$, which describes the rate of collision of a set of photons traveling through the medium. We distinguish absorbing ($\mu_\mathrm{a}$) and scattering ($\mu_\mathrm{s}$) particles, with the combined collision coefficient $\mu_\mathrm{t} = \mu_\mathrm{a} + \mu_\mathrm{s}$, also referred to as the extinction coefficient. The probability of a photon traveling through the medium from $\mathbf{x}$ to $\mathbf{y}$ without collision can be interpreted as a fractional visibility and is called the *transmittance* $T(\mathbf{x}, \mathbf{y})$:

$$T(\mathbf{x}, \mathbf{y}) = e^{-\tau(\mathbf{x},\mathbf{y})}, \tag{2.21}$$

$$\tau(\mathbf{x}, \mathbf{y}) = \int_0^{\|\mathbf{x}-\mathbf{y}\|} \mu_\mathrm{t}(\mathbf{x} + s\,\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}})\,\mathrm{d}s. \tag{2.22}$$

Here, $\tau$ is called the *optical depth*. In the case of a *homogeneous* medium, i.e. constant $\mu_\mathrm{t}$, this simplifies to $\tau(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|\,\mu_\mathrm{t}$. If the line between $\mathbf{x}$ and $\mathbf{y}$ is blocked by a surface, we set $T(\mathbf{x}, \mathbf{y}) = 0$.

**Light sources.** Surfaces or media where photon emission events occur are considered light sources. There are also some additional special cases of light sources. For example, light coming from the surrounding environment is usually modeled by an infinitely distant light source [BN76]. We assume explicit knowledge of light sources in a scene, i.e. it is possible to efficiently create emissive vertices in order to begin or terminate a path. An emission distribution function (EDF) describes the directional distribution of emitted radiance. In the measurement contribution function, radiance emitted from $\mathbf{x}$ toward $\mathbf{y}$ occurs as

$$L_\mathrm{e}^*(\mathbf{x}, \mathbf{y}) = \begin{cases} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}}) & \text{if } \mathbf{x} \text{ is on a surface,} \\ L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}})\,\mu_\mathrm{a}(\mathbf{x}) & \text{if } \mathbf{x} \text{ is in a medium,} \end{cases} \tag{2.23}$$

where we additionally multiplied by the absorption coefficient in emissive media.

**Material models.** Real-world materials exhibit various scattering and absorption behavior. We describe scattering by the macroscopic directional distribution of outgoing photon directions $\boldsymbol{\omega}_o$, conditional on an incident photon direction $\boldsymbol{\omega}_i$. This is called the bidirectional scattering[4] distribution function (BSDF) $f_\mathrm{r}(\boldsymbol{\omega}_o, \mathbf{x}, \boldsymbol{\omega}_i)$ $\left[\mathrm{sr}^{-1}\right]$ on surfaces, and the phase function $f_\mathrm{p}(\boldsymbol{\omega}_o, \mathbf{x}, \boldsymbol{\omega}_i)$ $\left[\mathrm{sr}^{-1}\right]$ in media. We unify both in

[4] In the absence of transmission, it is defined on the upper hemisphere only and called the bidirectional reflectance distribution function (BRDF).

the scattering function $f_\mathrm{s}$:

$$f_\mathrm{s}(\mathbf{x},\mathbf{y},\mathbf{z}) = \begin{cases} f_\mathrm{r}(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}},\mathbf{y},\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}}) & \text{if } \mathbf{y} \text{ is on a surface,} \\ f_\mathrm{p}(\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{x}},\mathbf{y},\boldsymbol{\omega}_{\mathbf{y}\to\mathbf{z}})\,\mu_\mathrm{s}(\mathbf{y}) & \text{if } \mathbf{y} \text{ is in a medium.} \end{cases} \tag{2.24}$$

An important special case of scattering is when $\boldsymbol{\omega}_o$ is completely determined by $\boldsymbol{\omega}_i$. These *specular* interactions occur for example on perfectly smooth reflecting or refracting surfaces, such as mirrors or glass. These are commonly written with a Dirac delta symbol ($\delta$) inside $f_\mathrm{s}$, although technically such a function does not exist. See Jakob [Jak13, Ch. 4] and Jung [Jun24, Sec. 4.8] for a more formal treatment. Materials that are close to specular, i.e. when the scattering distribution has narrow peaks, are called *glossy*. These make light transport challenging to simulate, as we discuss in Section 2.3.

Absorption on surfaces is modeled implicitly by a loss of energy in the BSDF, i.e. commonly $\int_\Omega f_\mathrm{r}(\boldsymbol{\omega}_o,\mathbf{x},\boldsymbol{\omega}_i)\,|n(\mathbf{x})\cdot\boldsymbol{\omega}_i|\,\mathrm{d}\boldsymbol{\omega}_i < 1$. The scattering albedo $\alpha = \mu_\mathrm{s}/\mu_\mathrm{t} = 1 - \mu_\mathrm{a}/\mu_\mathrm{t}$ is the equivalent in media, since the phase function is normalized by definition.

**Cameras.** The goal of our light transport simulation is to obtain virtual measurements on sensors, which we assume to be part of a virtual camera. The *pinhole* camera is the most basic model for a camera. Due to its infinitesimal aperture, incident directions map one-to-one to a point on the sensor. We use the *thin lens* model instead, which additionally requires a position on the aperture to map between directions and sensor points. This allows us to reproduce important effects like depth of field. More realistic modeling of cameras can be achieved by simulating light transport inside lens systems [KMH95] or using polynomial approximations [HD14]. We abstract from specific camera models using the *sensor responsivity* $W_j(\mathbf{x},\mathbf{y}) = W_j(\mathbf{x},\boldsymbol{\omega}_{\mathbf{x}\to\mathbf{y}})$, which also contains a reconstruction filter for pixel $j$.

**Extensions.** The basic model we have described is sufficient for many applications and in common use. Beyond the scope of this dissertation, many extensions of it have been explored in computer graphics research. For example, it can be important for some applications to additionally consider polarization [WW12] or fluorescence [Gla95]. Media can also be modeled with an anisotropic cross-section [JAM*10], continuous IOR [ABW14], and correlated particles [BRM*18; JAG18; dEon18]. These extensions can introduce additional challenges for efficient path sampling, but are still compatible with our fundamental framework of light transport.

### 2.1.6 Measurement contribution function

With the definitions from above, we can now define the measurement contribution function $f_j$ of a path with $k+1$ vertices with respect to the path space measure as the product

$$f_j(\bar{\mathbf{x}}) = W_j(\mathbf{x}_0,\mathbf{x}_1)\,L_\mathrm{e}^*(\mathbf{x}_k,\mathbf{x}_{k-1})\,G(\mathbf{x}_0,\mathbf{x}_1)\,T(\mathbf{x}_0,\mathbf{x}_1)\prod_{i=1}^{k-1} f_\mathrm{s}(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1})\,G(\mathbf{x}_i,\mathbf{x}_{i+1})\,T(\mathbf{x}_i,\mathbf{x}_{i+1}). \tag{2.25}$$

Each term can be associated with a vertex or an edge of the path (see Fig. 2.5). This completes our definition of the light transport problem, which is to compute the integral of Eq. (2.25) over path space

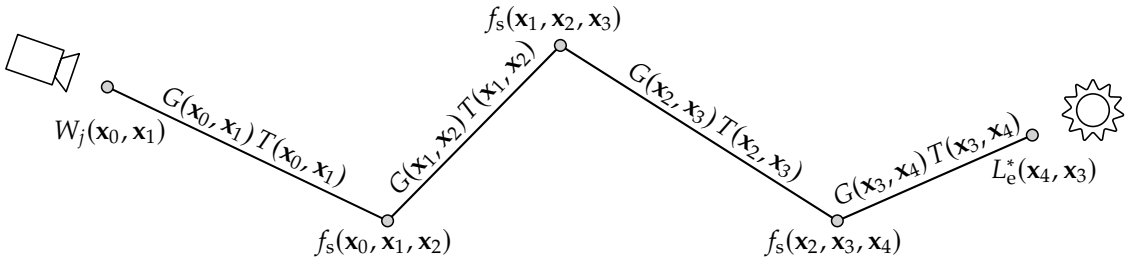



**Figure 2.5:** *Each term of the measurement contribution function is associated with a path vertex or edge. Here, we annotate the path* $\bar{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ *as an example.*

(see Eq. (2.2)). Before we discuss specific methods for light transport, we first outline the more general approach to estimating the value of such integrals using Monte Carlo integration.

## 2.2 Monte Carlo integration

Numerical integration of an equation like the measurement equation is challenging due to its high dimensionality and discontinuities in the integrand, which originate in the scene geometry. The fundamental idea of Monte Carlo integration is to construct a random variable whose expected value is equal to the integral of interest. By drawing samples of the random variable, we obtain an empirical estimate of the expected value. This approach does not require a smooth integrand and can scale well with dimensionality, provided we are able to obtain a reliable estimate efficiently. In this section, we first review fundamental definitions from probability theory in order to formalize the idea of Monte Carlo integration. Furthermore, we discuss approaches for improving efficiency and robustness of Monte Carlo estimators.

### 2.2.1 Random variables and expectation

A random variable is a measurable function $X\colon \mathcal{U} \to \mathbb{R}$ that maps an element from the *sample space* $\mathcal{U}$ to a *realization* of $X$. The sample space is part of the *probability space* $(\mathcal{U}, \sigma(\mathcal{U}), P)$, which in addition contains a $\sigma$-algebra on $\mathcal{U}$ and a *probability measure* $P$. A probability measure is a measure as defined in Section 2.1.2, with the additional requirement $P(\mathcal{U}) = 1$. For our purposes, we use the unit hypercube $\mathcal{U} = [0, 1)^k$ with Lebesgue measure $P = \lambda_k$.

**Distribution and probability density (PDF).** The push-forward measure $P \circ X^{-1}$ (see Eq. (2.6)) of a random variable $X$ defines a new probability measure, also referred to as its *distribution*. Its Radon-Nikodym derivative with regard to a reference measure $\mu$ is the *probability density* (PDF)

$$p_X = \frac{\mathrm{d}P \circ X^{-1}}{\mathrm{d}\mu}. \tag{2.26}$$

If the PDF exists, it inherits the normalization property of the probability measure:

$$\int_{X(\mathcal{U})} p_X(x)\,\mathrm{d}\mu(x) = \int_{X(\mathcal{U})} \mathrm{d}P \circ X^{-1}(x) = \int_{\mathcal{U}} \mathrm{d}P(u) = P(\mathcal{U}) = 1. \tag{2.27}$$

**Expectation.** The *expected value* $\mathbb{E}[X]$ of a random variable $X$ is defined by

$$\mathbb{E}[X] = \int_{\mathcal{U}} X(u)\,\mathrm{d}P(u). \tag{2.28}$$

With a change of variables, we can also express it using a PDF if it exists:

$$\mathbb{E}[X] = \int_{X(\mathcal{U})} x\,\mathrm{d}P \circ X^{-1}(x) = \int_{-\infty}^{\infty} x\,p_X(x)\,\mathrm{d}\mu(x). \tag{2.29}$$

It is often convenient to define another random variable by $Y = g(X) = g \circ X$, where $g$ is a measurable function, and express the expected value of $g(X)$ in terms of the distribution of $X$:

$$\mathbb{E}\left[g(X)\right] = \int_{X(\mathcal{U})} g(x)\,\mathrm{d}P \circ X^{-1}(x). \tag{2.30}$$

**Variance and covariance.** Another important property of a random variable is *variance*. It describes the "spread" of a random variable and is defined as the expected squared deviation from its expectation:

$$\operatorname{var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]. \tag{2.31}$$

This definition can be generalized to the *covariance* of two random variables $X$ and $Y$

$$\operatorname{cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}[X])\,(Y - \mathbb{E}[Y])\right]. \tag{2.32}$$

We call $X$ and $Y$ *uncorrelated* when $\operatorname{cov}(X, Y) = 0$. In this case, $\mathbb{E}[X\,Y] = \mathbb{E}[X]\,\mathbb{E}[Y]$.

**Joint distribution.** To reason more precisely about the distribution of multiple random variables, we define a *random vector* $\bar{X} \colon \mathcal{U} \to \mathbb{R}^k, u \mapsto (X_1(u), \dots, X_k(u))$, where $X_1, \dots, X_k$ are random variables. The *joint distribution* is then defined as the push-forward measure of $\bar{X}$, i.e. $P \circ \bar{X}^{-1}$. In this context, we refer to the individual distributions $P \circ X_1^{-1}, \dots, P \circ X_k^{-1}$ as *marginal* distributions. $X_1, \dots, X_k$ are called *independent* if their joint distribution is the product of marginal distributions:

$$P \circ \bar{X}^{-1} = P \circ X_1^{-1} \times \cdots \times P \circ X_k^{-1}. \tag{2.33}$$

We also define the joint and marginal PDF as the density of the respective distributions. Similar to Eq. (2.33), iff $X_1, \dots, X_k$ are independent, their joint PDF is the product of marginal PDFs.

**Conditional probability.** Conditional probability allows us to reason about probabilities under partial knowledge. Consider the probability measure $P$ and events $A, B \in \sigma(\mathcal{U})$. The conditional probability of

$A$ given $B$ is defined by

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}. \tag{2.34}$$

For a random variable $X$, its *conditional expectation* $\mathbb{E}[X \mid \mathcal{G}]$ given a sub-$\sigma$-algebra $\mathcal{G} \subseteq \sigma(\mathcal{U})$ satisfies

$$\int_A \mathbb{E}[X \mid \mathcal{G}] \, \mathrm{d}P = \int_A X \, \mathrm{d}P, \quad \text{for any } A \in \mathcal{G}. \tag{2.35}$$

This definition can then be used to define conditional distributions, although we do not require the full generality of this definition for our purposes. Instead, we simplify by using probability densities, since we can often assume their existence in our application. We define the conditional PDF $p_{X|Y}$ in terms of the ratio of joint PDF $p_{X,Y}$ and marginal PDF $p_Y$, where $X$ and $Y$ are random variables:

$$p_{X|Y}(x \mid y) = \frac{p_{X,Y}(x, y)}{p_Y(y)}. \tag{2.36}$$

We can also write $p_{X|Y}(x \mid Y)$, turning the PDF into a function of $Y$ and therefore a random variable. The marginal PDF is the expectation of this random variable, which relates the marginal to the joint PDF:

$$p_X(x) = \mathbb{E}\left[p_{X|Y}(x \mid Y)\right] = \int_{-\infty}^{\infty} p_{X|Y}(x \mid y) \, p_Y(y) \, \mathrm{d}y = \int_{-\infty}^{\infty} p_{X,Y}(x, y) \, \mathrm{d}y. \tag{2.37}$$

The conditional expectation of $X$ given $Y = y$ can be defined in terms of the conditional PDF by

$$\mathbb{E}[X \mid Y = y] = \int_{-\infty}^{\infty} x \, p_{X|Y}(x \mid y) \, \mathrm{d}x. \tag{2.38}$$

Similar to the case above, we also make use of the notation $\mathbb{E}[X \mid Y]$, where the conditional expectation is a function of $Y$ and therefore a random variable itself.

### 2.2.2 Monte Carlo estimator

To compute the value of the integral $I = \int_\Omega f(x) \, \mathrm{d}\mu(x)$, we define the primary Monte Carlo estimator

$$\hat{F} = \frac{f(X)}{p(X)}, \tag{2.39}$$

where $X$ is a random variable with probability density $p$ with respect to $\mu$, and $X(\mathcal{U}) = \Omega$. Since $p(x) = \frac{\mathrm{d}P \circ X^{-1}}{\mathrm{d}\mu}$, the expected value of $\hat{F}$ is equal to the integral of interest:

$$\mathbb{E}\left[\hat{F}\right] = \mathbb{E}\left[\frac{f(X)}{p(X)}\right] = \int_{\mathcal{U}} \frac{f(X(u))}{p(X(u))} \, \mathrm{d}P(u) = \int_{X(\mathcal{U})} \frac{f(x)}{p(x)} \, \mathrm{d}P \circ X^{-1}(x) = \int_\Omega f(x) \, \mathrm{d}\mu(x) = I. \tag{2.40}$$

We refer to an estimator with this property as being *unbiased*. Convergence is achieved using a secondary estimator $\hat{F}_n$ that averages $n$ independent estimates:

$$\hat{F}_n = \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{p(X_i)}. \tag{2.41}$$

Due to the linearity of expectation, $\hat{F}_n$ remains unbiased. It is also *consistent*, i.e. it converges to the correct value in the limit, since $\hat{F}_n$ tends toward its expectation because of the law of large numbers:

$$\hat{F}_n \xrightarrow{\text{a.s.}} \mathbb{E}[\hat{F}_n] = I, \text{ when } n \to \infty. \tag{2.42}$$

**Expected error.** Since $\hat{F}_n$ is unbiased, its expected squared error (MSE) is equal to its variance

$$\text{MSE} = \mathbb{E}\left[\left(\hat{F}_n - I\right)^2\right] = \mathbb{E}\left[\left(\hat{F}_n - \mathbb{E}\left[\hat{F}_n\right]\right)^2\right] = \text{var}\left(\hat{F}_n\right), \tag{2.43}$$

and relates to the variance of the primary estimator by the reciprocal of the number of samples

$$\text{var}\left(\hat{F}_n\right) = \text{var}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}\right) = \frac{1}{n^2}\,\text{var}\left(\sum_{i=1}^{n}\frac{f(X_i)}{p(X_i)}\right) = \frac{1}{n^2}\sum_{i=1}^{n}\text{var}\left(\hat{F}\right) = \frac{1}{n}\,\text{var}\left(\hat{F}\right). \tag{2.44}$$

The root mean squared error (RMSE) is often a more natural error metric and equal to standard deviation:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\,\text{var}\left(\hat{F}\right)} = \frac{\sigma\left[\hat{F}\right]}{\sqrt{n}}. \tag{2.45}$$

We therefore say that Monte Carlo integration converges at a rate of $O\left(n^{-0.5}\right)$, where the constant is the standard deviation of the primary estimator. To reduce the error, we can simply take more random samples with a proportional increase in required computation. However, since to halve the error we have to take four times as many samples, we quickly run into diminishing returns. If estimator variance is too high, achieving convergence in this way becomes impractical.

**Variance reduction.** Reducing estimator variance is often essential to bringing down the error of Monte Carlo integration. There exist multiple approaches to reduce different sources of variance, e.g. stratification, antithetic sampling or control variates [Vea97]. We discuss *importance sampling*, which is a particularly relevant technique for reducing variance in light transport simulation, in the next section.

### 2.2.3 Importance sampling

Importance sampling is a technique to reduce the variance of $\hat{F}$ by making the ratio $f(X)/p(X)$ more uniform. Consider for example the case when the integrand $f$ has a high peak that is not reflected in $p$. If we place a sample $x'$ inside the region of the peak, the ratio $f(x')/p(x')$ will be comparatively high. Since this case occurs infrequently, $x'$ will be an outlier sample and increase variance substantially. Instead, we would ideally choose $p$ proportional to $f$, leading to $f(X)/p(X) = I$ and zero variance. This would however imply knowing $I$ in advance, defeating the purpose of Monte Carlo integration.

**Inverse transform sampling.** Typically, we will instead choose an approximation $\tilde{f} \approx f$, for which we are able to obtain a random variable $X$ with probability density $p \sim \tilde{f}$. Inverse transform sampling is a method to construct such a random variable with given distribution. Let $X'$ be a random variable with the targeted probability density, i.e. $p_{X'}(x) = \tilde{f}(x)/\left(\int_{-\infty}^{\infty}\tilde{f}(y)\,\mathrm{d}y\right)$. The cumulative distribution function

$F_{X'}$ and its (generalized) inversion, the quantile function $F^{-1}_{X'}$, are defined by

$$\begin{aligned} F_{X'}(x) &= P \circ X'^{-1}((-\infty, x]) = \int_{-\infty}^{x} p_{X'}(y)\,\mathrm{d}y, \\ F^{-1}_{X'}(u) &= \inf\{x \in \mathbb{R} : u \leq F_{X'}(x)\}. \end{aligned} \tag{2.46}$$

We can now define a random variable $X\colon [0,1) \to \mathbb{R}$ via the quantile function of $X'$:

$$X(u) = F^{-1}_{X'}(u). \tag{2.47}$$

$X$ has the same distribution as $X'$, and therefore the targeted probability density. In summary, to apply inverse transform sampling, we first integrate the desired probability density and then invert the resulting distribution function. To use inverse transform sampling with a higher dimensional probability density $q$, we first factorize it, e.g. $q(x,y) = q(x \mid y)\, q(y)$, and then apply the method to each factor individually. For the approximation $\tilde{f}$, it is often a viable choice to omit some factors of $f$ that would make inverse transform sampling intractable. In our application this is convenient, because our integrand $f$ is defined as a product.

**Multiple importance sampling (MIS).** In many cases, there may be several candidates for factors to importance sample, resulting in different sampling *techniques*. While each technique may be a good fit for a certain region of the domain, samples outside of it still cause high variance. It is often difficult to cleanly separate these regions, and simply adding estimates with high variance will only increase it further. *Multiple importance sampling* [VG95b] provides a more careful combination of $m$ sampling techniques. We first draw $n_i$ samples for each $i$-th technique, and compute a weighted average:

$$\hat{F}_{\mathrm{MIS}} = \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})}. \tag{2.48}$$

The estimate is unbiased, as long as the weighting function $w$ fulfills the following conditions:

**(W1)** $\sum_{i=1}^{m} w_i(x) = 1$ when $f(x) \neq 0$,

**(W2)** $w_i(x) = 0$ when $p_i(x) = 0$.

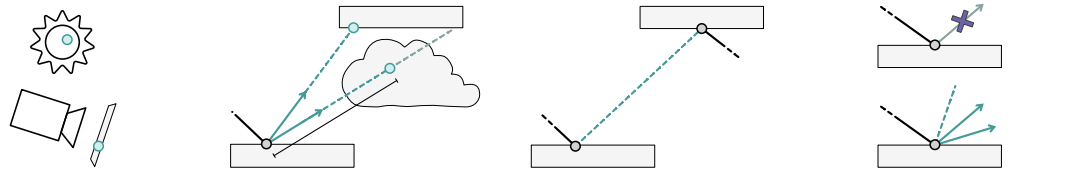
A robust, low-variance choice is the *balance heuristic*

$$w_i(x) = \frac{n_i p_i(x)}{\sum_{k=1}^{m} n_k p_k(x)}. \tag{2.49}$$

We can interpret it as (stratified) sampling from a mixture of techniques, which becomes more apparent when we insert it as the weight into Eq. (2.48):

$$\sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \frac{f(X_{i,j})}{\sum_{k=1}^{m} n_k p_k(X_{i,j})}. \tag{2.50}$$

There is also the *one-sample model* of MIS, where we randomly choose a technique $i$ for each sample proportional to a probability $c_i$, instead of drawing a predetermined number of samples $n_i$ for each



**(a)** *Vertex sampling* **(b)** *Path extension* **(c)** *Vertex connection* **(d)** *Russian roulette (top) and splitting (bottom)*

**Figure 2.6:** *Path sampling algorithms are constructed using the shown basic operations. Vertices can be sampled on light sources or cameras (a). A subpath can be extended by an additional vertex (b), or connected to a different subpath (c). Subpaths can also be terminated or duplicated (d).*

technique. This is useful when there are more techniques available than samples we can afford to draw. In this case, balance heuristic weights produce the lowest variance for given $c_i$.

## 2.3 Transport path sampling

Our general approach to light transport simulation is to estimate the measurement equation (Eq. (2.2)) using Monte Carlo integration with multiple importance sampling. Effectively, this transforms the integration problem into one of sampling paths. Path sampling algorithms fundamentally determine the efficiency of light transport simulation through the variance of the Monte Carlo estimator. It is difficult to find a general solution to this problem that can handle every kind of scene well. Even for algorithms that are designed for dealing with complicated transport, there remain cases where they fail. These often introduce significant overhead compared to more lightweight algorithms in simpler scenes. Consequently, various approaches to improving path sampling have been explored, and this continues to be an active topic of research.

In this section, we outline the state of the art in path sampling algorithms. We introduce common path sampling operations, combine them into basic algorithms, and discuss their limitations and general challenges for efficient sampling. We conclude with an overview of more advanced approaches, such as Markov chain Monte Carlo or path guiding.

### 2.3.1 Path sampling operations

The following set of basic operations forms the foundation of path sampling algorithms (also see Fig. 2.6).

**Vertex sampling.** We randomly sample an unconnected vertex on a light source, or on the image plane of a camera, to start a new path. Depending on its origin, we refer to an incomplete path as a camera or light subpath. Specific sampling methods depend on light source and camera models. For example, if using a thin lens camera, we would typically sample a position on the image plane and a position on the aperture to obtain the outgoing direction. To begin a light subpath, we select a light source first, typically in proportion to flux. Then we sample a vertex on its surface and an outgoing direction in proportion to the directional distribution of emission.

**Path extension.** To sample longer paths, we extend a subpath with additional vertices one by one. Starting from the previous path vertex, we first sample an outgoing direction and place the new vertex at the next intersection with the scene. We usually importance sample the local scattering function $f_s$ if possible, i.e. the BSDF or phase function. The intersection with the scene can be either with a surface or with a participating medium. To intersect a medium, we usually sample the distance to next collision (*free flight*) in proportion to transmittance. We place the vertex at the closest intersection, either in the medium or on a surface. It is also possible to stochastically choose between medium or surface intersection a priori, decoupling this decision from transmittance [VWF18].

**Vertex connection.** Another option is to connect two different vertices, usually to form a complete path out of subpaths. An important example for this is *next-event estimation* [CCY67], where we connect a camera subpath to a vertex sampled on a light source. Commonly, we only connect separately sampled subpaths, since this allows us to evaluate the PDF of the resulting path as the product of subpath PDFs. Other constructions often lead to an intractable path PDF. For example, "skipping" a vertex on a path by connecting its predecessor to its successor would require solving an integral for the marginal PDF of the two remaining vertices.

**Russian roulette and splitting.** Eventually, we have to finish extending a path with further vertices, which would happen in the photon model on an absorption event. In light transport simulation, it is common to always extend a path by a further scattering event instead, unless the material is fully absorbing, which is typically a light source. Absorption is instead accounted for in the sampling weight, i.e. the Monte Carlo estimate. This approach is known as *implicit capture*.

To avoid sampling overly long paths, the path length is often limited to a maximum number of vertices in practice. This, however, systematically neglects higher-order transport, even though the additional contribution is often negligible. *Russian roulette* is an alternative that terminates paths probabilistically and was introduced to graphics by Arvo [AK90]. Its main idea is to construct an estimator $\hat{F}'$ as follows:

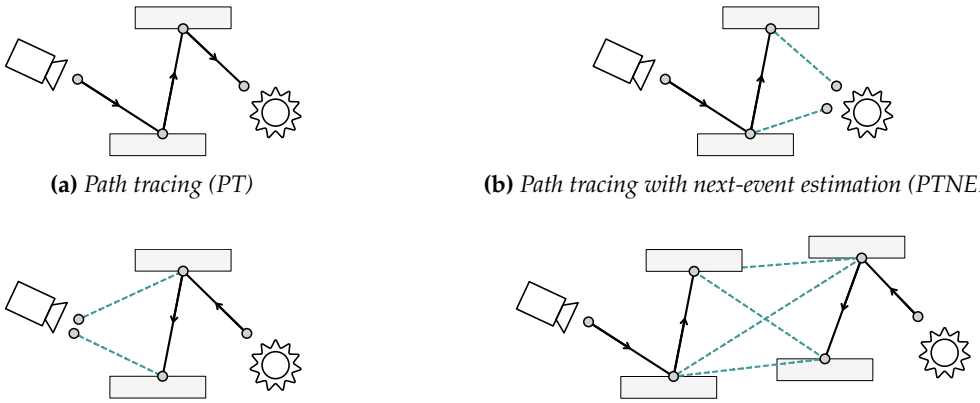
$$\hat{F}' = \begin{cases} \frac{\hat{F}}{P} & \text{with survival probability } P, \\ 0 & \text{otherwise.} \end{cases} \tag{2.51}$$

With termination probability $1 - P$, we do not have to evaluate $\hat{F}$, but the estimate remains unbiased:

$$\mathbb{E}\left[\hat{F}'\right] = P\,\mathbb{E}\left[\frac{\hat{F}}{P}\right] + (1-P)\,\mathbb{E}[0] = \mathbb{E}\left[\hat{F}\right]. \tag{2.52}$$

If $\hat{F}$ is very expensive to evaluate, $\hat{F}'$ can be more efficient despite its increased variance. This can be applied recursively to terminate paths. Choosing the survival probability based on local quantities alone, e.g. surface albedo, can increase variance significantly. A more robust probability can be obtained by estimating the expected contribution of a subpath [VK16]. Rath et al. [RGH*22] adjust the termination probability by optimizing for the efficiency of the estimator explicitly.

It is also possible to *split* a subpath, i.e. to duplicate it and continue sampling for each duplicate independently. Because of the shared prefix, it results in correlated samples. While these possibly converge more slowly than independent samples, they still maintain the correct expectation of the



**(a)** *Path tracing (PT)*

**(b)** *Path tracing with next-event estimation (PTNEE)*

**(c)** *Light tracing (LT)*

**(d)** *Bidirectional path tracing (BDPT)*

**Figure 2.7:** *Simple path tracing (a) constructs paths by iteratively extending a subpath starting from the camera, until encountering a light source by chance. It can be enhanced by connecting to vertices on light sources (b). Paths can also be traced starting from a light source (c). Both methods can be combined by forming inner connections between a camera and light subpath (d).*

estimator. Probabilistic splitting can be seen as the opposite of Russian roulette, as it reduces variance but increases computational overhead. Therefore, similar algorithms can be employed for choosing the splitting probability [VK16; RGH*22].

### 2.3.2 Path sampling algorithms

The sampling operations from the previous section can be combined in different ways, leading to the following common path sampling algorithms (shown in Fig. 2.7).

**Path tracing (PT).** The simplest approach to path sampling is to start a path at the camera and then extend it vertex by vertex, until we randomly encounter a light source [Kaj86]. This has the advantage that we can often importance sample almost all terms of the measurement equation well. However, the important emission term is not importance sampled, since it depends on random intersection of the light source. Consequently, the variance of this sampling technique can be arbitrarily high with small light sources. It can, however, work very well where light sources subtend a large solid angle. We also refer to this sampling algorithm as *analog* path tracing, since it follows the (reversed) random walk of a photon as described by the light transport model.

**Path tracing with next-event estimation (PTNEE).** We can extend simple path tracing with next-event estimation. At each path vertex, in addition to the path extension, we also sample a light source vertex and connect to it [Kaj86]. This improves sampling of small light sources. It also amortizes the construction of the path prefix, since we construct multiple correlated paths via splitting, without increasing the number of sampled paths exponentially. Unfortunately, connections are not possible at specular materials and become ineffective for highly glossy ones. Especially in media, the weak singularity of the $G$ term is problematic and can even lead to infinite variance [KF12]. Nevertheless, path tracing with next-event estimation is the most widely used path sampling technique in practice.

Its efficiency can be improved by careful importance sampling of the light vertex, e.g. through light hierarchies [PPD98; EK18] or sampling proportional to (projected) solid angle [Arv95a; Arv01] and the BSDF or phase function [DHB17; Pet21].

**Light tracing (LT).** A complementary technique to path tracing is to start at a light source and trace paths toward the camera [Arv86; DLW93]. This seems less efficient at first, since we lose the implicit focus on visible parts of the scene that we get from starting at the camera. However, light tracing is able to efficiently sample caustic paths, i.e. indirect illumination with specular interactions before a small light source, which would likely be missed by path tracing. Light tracing is typically only used with next-event connections to the camera, since it would be highly unlikely to randomly intersect the camera sensor. An advantage of this is that paths can contribute to the estimate of multiple pixels, which is especially helpful in media. Since light tracing is still susceptible to the problems common to all techniques based on explicit connections, it cannot sample paths with directly visible specular surfaces.

**Bidirectional path tracing (BDPT).** To unify both approaches, they can be combined into bidirectional path tracing [LW93; VG95a]. We first sample a camera and a light subpath with $t$ and $s$ vertices, respectively. Then, we connect both subpaths in every possible way, resulting in $O(t\,s)$ paths. Next-event connections are additionally formed while tracing the subpaths, without these paths participating in the inner connections. We end up with many available techniques for sampling a path and achieve a robust estimate by combining them using MIS [VG95a]. However, in many simple cases, path tracing with next-event estimation is already an effective technique in practice. As BDPT comes with a high overhead, it can be uncompetitive if none of its sampling techniques offers much lower variance than more lightweight algorithms.

### 2.3.3 Challenges

In many common scene configurations, the previously discussed path sampling algorithms already achieve low variance estimates. However, even the many techniques of BDPT are not able to sample every kind of path with sufficient density. Therefore, it is easy to find examples for configurations where light transport can only be estimated with high variance.

**Light path expression.** We distinguish different categories of paths by classifying each of their interactions as either specular (`S`) or diffuse (`D`), and map the sequence of interactions of each path to a string of symbols. To indicate the vertex order, the string is enclosed by the symbols `L` for the light source, and `E` for the sensor. A regular expression matching these strings defines a category of paths [Hec90]. For inner vertices, the classification is based on the material model. We classify an interaction as `S` when there is a deterministic relation between incident and outgoing direction, and as `D` otherwise. Sensors and light sources can be described by a tuple `(S|D)(S|D)`, where the symbols specify finite (`D`) or infinitesimal (`S`) extent [Vea97, Sec. 8.3]. The outer symbol, i.e. the one next to `L` or `E`, refers to the spatial dimensions, while the inner one describes the directional dimensions. This categorization is very coarse, as for example glossy materials or small light sources are classified `D`, even though they can be close to being specular or infinitesimal (`S`). We should therefore consider `S` as the limiting case, and already expect problems related to `S` vertices to occur in practice when we approach it.

**Limitations of local sampling.** We refer to any algorithm that only uses the operations described in Section 2.3.1 as *local* sampling. Veach [Vea97, Sec. 8.3] shows that local sampling is only able to sample paths that contain the string `DD`. This is because to form a complete path, we either have to intersect the sensor or light source by chance, or connect two vertices explicitly. Connections require two consecutive `D` in the directional domain, as otherwise the connection will have zero contribution. Intersection of either sensor or light source requires them to have finite spatial extent, and also finite directional extent for a non-zero contribution. This theoretical limitation is not as severe in practice as it appears at first, since physical light sources are always `DD`. However, in this case we rely on the simple path tracing technique (PT) only, which often has high variance in practice. This is because, as discussed above, paths that would be impossible to sample with an `S` interaction are increasingly unlikely to be sampled as we approach the limiting case.

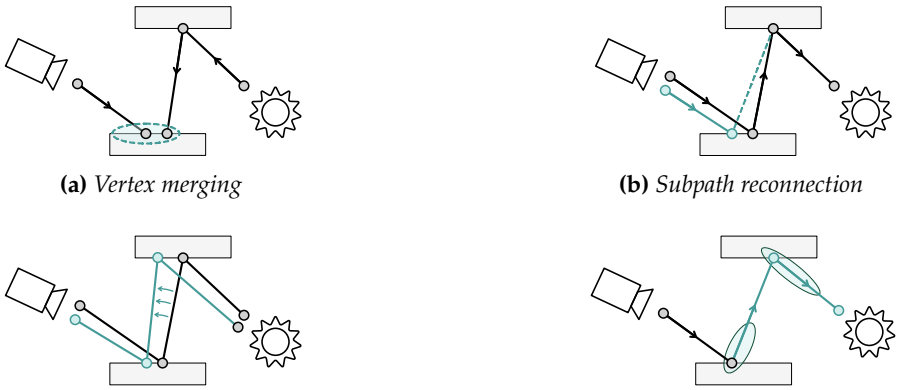
**Remedies for specular transport.** One way to deal with these problems is to *regularize* specular transport by artificially replacing specular materials with glossy ones [KD13]. This allows us to form valid connections between vertices where impossible prior to regularization. Additional path sampling techniques become available and variance is reduced. However, since we modify scene parameters, we also introduce bias and arrive at an incorrect result. With limited computation time, this tradeoff between bias and variance can still be worthwhile if applied with consideration of visual impact [WDH*21].

Another approach is to escape the limitations of local sampling by creating additional vertices when forming connections. We refer to this approach as *extended connections*. For example, the additional degrees of freedom introduced by the extra vertices allow us to explicitly solve specular constraints by root finding [HDF15; ZGJ20]. These techniques can resolve theoretical limitations of path sampling and are efficient for special cases in practice. However, they are also computationally expensive and require selective application in order to amortize.

**Additional challenges.** The inability to retain efficiency in simple cases is a general problem of many more advanced path sampling algorithms. BDPT is able to robustly sample various kinds of paths using its many techniques. However, if a path is best sampled by techniques corresponding to PT or PTNEE, we still pay the significant overhead of BDPT. More generally, while combining many techniques via MIS achieves robustness of the combined estimator, it does not solve the problem of reasonably allocating computational resources. While work toward explicitly optimizing this allocation exists, it is susceptible to noisy estimates [MRYS24]. Moreover, individual techniques are still limited by the information available to them. Light tracing, for example, usually excels in rendering directly visible caustics. However, since it does not consider visual importance, it might rarely intersect small caustic casters by chance, or distribute its samples sparsely in the view frustum in large scenes. Complex visibility or the additional degrees of freedom in media similarly make it challenging to focus sampling effort on important scene regions. In general, the local information available to path sampling is often insufficient for making sampling decisions that lead to a good distribution of sampled paths.

### 2.3.4 Advanced path sampling

A common trait of the algorithms discussed so far is that paths are used only once. Even if a difficult path is discovered by chance, there is no attempt to explore similar paths. While PTNEE or BDPT



**(a)** *Vertex merging* **(b)** *Subpath reconnection*

**(c)** *Markov chain Monte Carlo* **(d)** *Path guiding*

**Figure 2.8:** *More advanced sampling algorithms better utilize previously sampled paths. Kernel density estimation can be expressed in the path sampling framework as vertex merging (a). New paths can also be formed by reconnecting to suffixes of paths (b). Markov chain methods allow path sampling through mutation (c). Path guiding adaptively learns distributions for importance sampling (d).*

employ some exploration via splitting, their reuse of subpaths remains limited to the current sample only. Since we estimate many correlated integrals at once, i.e. one for each individual pixel, and we typically require many samples for convergence, significant potential is left unutilized. In this section, we discuss approaches for leveraging sampled subpaths more effectively by sharing information globally (also see Fig. 2.8).

**Vertex merging.** One way to achieve reuse is to store vertices of many light subpaths in a global data structure. On camera subpaths, we can then *merge* a vertex with a nearby light subpath vertex, corresponding to a kernel density estimate. This approach is traditionally known as *photon mapping* [Jen96]. With the interpretation of the density estimate as stochastically accepting a merge of two vertices, it can be imported into the path sampling framework [GKDS12; HPJ12]. It augments BDPT with the ability to sample previously unsampleable `SDS` paths efficiently, i.e. where connections are not possible due to a missing `DD` segment. The approach extends to participating media [KGH*14], where additional ways of performing merges need to be considered [JNSJ11]. Combining all these techniques has similar properties to those discussed before: we obtain a very robust estimator with potentially large overhead. It is possible to select a subset of techniques based on estimates of their efficiency, but the optimization can be sensitive to noise [GYGS22]. Moreover, even this many techniques are not able to cover all cases, as for example chains of glossy interactions remain difficult to sample, and photon density can be insufficient in visually important areas. In addition, the merge kernel introduces bias, which can be objectionable, especially on fine geometry such as hair.

**Unbiased subpath reuse.** Connections can also be employed for global reuse, thereby avoiding bias, while improving amortization of difficult subpaths. With BDPT for example, we can store light subpaths in a global data structure and sample one to connect to based on estimated contribution [PRDD15], similar to the use of light hierarchies. Taking the BSDF into account via an approximation can make connections on glossy surfaces more effective [TH19]. While these methods mitigate some of the

problems of BDPT, they also increase its overhead, especially in simple scenes.

It is also possible to connect newly sampled camera subpaths to the suffixes of previously sampled paths with unidirectional path tracing from the camera [BSH02; THD17]. This is a version of splitting, where we duplicate the path suffix instead of the prefix. Although conceptually similar, its application is more restricted, since a marginal probability density has to be evaluated when the camera subpaths are not chosen deterministically. This can be sidestepped using stochastic MIS [WGGH20].

Shift mappings [LKL*13; HGP*19] enable the reuse of paths from different sampling domains, e.g. neighboring pixels. Reservoir-based spatio-temporal importance resampling (ReSTIR) [BWP*20; LKB*22] chains multiple passes of probabilistic reuse to amplify the number of potentially contributing candidate paths. Visually distracting correlation artifacts can persist in difficult scenes due to insufficient candidates, which is a limitation shared by all methods that reuse subpaths. With ReSTIR, artifacts can be reduced by recombining prefix and suffix subpaths [KLR*23], or by mutating reused paths to distribute the error more evenly [SLK*24]. In general, while reuse helps with maximizing the utility of rarely sampled paths, it does not address the problem of reliably sampling independent paths.

**Markov chain Monte Carlo.** To facilitate local exploration of difficult regions in path space, we can sample a new path in the neighborhood of the previous one through mutation. This type of conditional sampling becomes possible by modeling the process of path sampling with a Markov chain. The correct sampling distribution is reached in the limit using the Metropolis-Hastings algorithm [Has70] for its transition. Using the path space as a state space for the Markov chain allows us to design mutations tailored to specific light transport problems [VG97]. Working with the primary sample space makes use of existing importance sampling techniques instead [KSAC02]. The latter approach greatly simplifies implementation. Making the selected MIS technique part of the Markov chain state further enhances the efficiency of mutations [HKD14]. It is also possible to work in the half-vector domain [JM12; KHD14; HKD15]. This allows us to explicitly express and solve for constraints introduced by specular and glossy interactions, improving sampling of respective difficult paths. Mutations defined in different spaces can be combined, although some care is required to define unambiguous mappings between path space and primary sample space [Pan17; OKH*17; BJNJ17]. We refer to the survey by Šik and Křivánek [ŠK18] for a wider overview of Markov chain methods in light transport simulation.

With all of these methods, it is difficult to achieve global stratification, since sample placement is controlled by the Markov chain. Therefore, PTNEE tends to be more efficient in simple scenes, due to its more even sample placement. With more difficult scenes, Markov chains can get "stuck" on a path if there is no suitable mutation available. Markov chains in general produce correlated samples that lead to more structured artifacts compared to the uniform noise produced by independent sampling. Because of the resulting flicker, these methods are usually unsuitable for rendering animations.

**Path guiding.** Another approach is to use the information gained from previous path sampling to adapt the importance sampling distribution, thereby focusing sampling effort on the important regions. These methods are summarized under the term *path guiding* [VHH*19; HSRM25]. While this idea was explored early on already [Jen95; LW95], later refinements in representations and learning algorithms led to its more widespread adoption [VKŠ*14; MGN17; RHL20]. Path guiding makes the missing global information accessible to local importance sampling. It can augment relatively simple algorithms like

PTNEE, extending their capabilities to more difficult transport. Otherwise, bidirectional sampling would typically be required, which can have significant overhead and can be difficult to combine with tricks applied in practice. Path guiding therefore has many desirable properties for practical light transport simulation. It does however require expressive representations for sampling distributions. These have to support efficient importance sampling and be learnable from sparse and noisy path samples at the same time. Sampling in media increases the dimensionality, making this even more challenging. We discuss the state of the art in path guiding in detail in Section 3.2.1.

## 2.A Derivation of solid angle density

The density of solid angle with respect to surface area can be understood intuitively using geometry. Our goal in this section is to verify this intuition using a more algebraic approach.

Let $\mathbf{x}(u,v) \in \mathbb{R}^3$ be a parametric surface with tangents $\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}$, $\mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}$ and normal $\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}$. We project to the unit sphere using $\boldsymbol{\omega}(u,v) = \frac{\mathbf{x}(u,v)}{r}$, where $r = \|\mathbf{x}(u,v)\|$ and $\boldsymbol{\omega}_u = \frac{\partial \boldsymbol{\omega}}{\partial u}$, $\boldsymbol{\omega}_v = \frac{\partial \boldsymbol{\omega}}{\partial v}$. From Eq. (2.11), we know $\frac{\mathrm{d}A}{\mathrm{d}(u,v)} = \|\mathbf{x}_u \times \mathbf{x}_v\|$ and $\frac{\mathrm{d}\sigma}{\mathrm{d}(u,v)} = \|\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v\|$. Therefore, the unknown density is

$$\frac{\mathrm{d}\sigma}{\mathrm{d}A} = \frac{\|\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v\|}{\|\mathbf{x}_u \times \mathbf{x}_v\|}. \tag{2.53}$$

To get to the more familiar equation, we first make the derivative $\boldsymbol{\omega}_u$ explicit

$$\boldsymbol{\omega}_u = \frac{\partial \boldsymbol{\omega}}{\partial u} = \frac{\partial}{\partial u}\frac{\mathbf{x}}{r} = \frac{r \cdot \frac{\partial \mathbf{x}}{\partial u} - \mathbf{x}\frac{\partial r}{\partial u}}{r^2} = \frac{\mathbf{x}_u}{r} - \frac{(\mathbf{x}_u \cdot \mathbf{x})\mathbf{x}}{r^3} = \frac{1}{r}(\mathbf{x}_u - (\mathbf{x}_u \cdot \boldsymbol{\omega})\,\boldsymbol{\omega}), \tag{2.54}$$

and observe that it is orthogonal to $\boldsymbol{\omega}$:

$$\boldsymbol{\omega}_u \cdot \boldsymbol{\omega} = \frac{1}{r}\left(\mathbf{x}_u \cdot \boldsymbol{\omega} - (\mathbf{x}_u \cdot \boldsymbol{\omega})\,\boldsymbol{\omega} \cdot \boldsymbol{\omega}\right) = \frac{1}{r}\left(\mathbf{x}_u \cdot \boldsymbol{\omega} - \mathbf{x}_u \cdot \boldsymbol{\omega}\right) = 0. \tag{2.55}$$

Since the same equations apply when exchanging $u$ for $v$, $\boldsymbol{\omega}$ must be parallel to $\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v$. Since $\boldsymbol{\omega}$ has unit length, we can express the length of $\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v$ as its dot product with $\boldsymbol{\omega}$:

$$\begin{aligned}
\|\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v\| &= |\boldsymbol{\omega} \cdot (\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v)| && \text{(2.56a)}\\
&= \left|\det\begin{bmatrix}\boldsymbol{\omega} & \boldsymbol{\omega}_u & \boldsymbol{\omega}_v\end{bmatrix}\right| && \text{(2.56b)}\\
&= \frac{1}{r^2}\left|\det\begin{bmatrix}\boldsymbol{\omega} & \mathbf{x}_u & \mathbf{x}_v\end{bmatrix}\right| && \text{(2.56c)}\\
&= \frac{|\boldsymbol{\omega} \cdot (\mathbf{x}_u \times \mathbf{x}_v)|}{r^2} && \text{(2.56d)}\\
&= \frac{|\boldsymbol{\omega} \cdot (\|\mathbf{x}_u \times \mathbf{x}_v\|\,\mathbf{n})|}{r^2} = \frac{|\boldsymbol{\omega} \cdot \mathbf{n}|\,\|\mathbf{x}_u \times \mathbf{x}_v\|}{r^2}. && \text{(2.56e)}
\end{aligned}$$

Here, we used the identity $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \det\begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{bmatrix}$, Gaussian elimination, and finally inserted the definition of the surface normal. Inserting this result into Eq. (2.53), we arrive at

$$\frac{\mathrm{d}\sigma}{\mathrm{d}A} = \frac{\|\boldsymbol{\omega}_u \times \boldsymbol{\omega}_v\|}{\|\mathbf{x}_u \times \mathbf{x}_v\|} = \frac{|\boldsymbol{\omega} \cdot \mathbf{n}|\,\|\mathbf{x}_u \times \mathbf{x}_v\|}{r^2\,\|\mathbf{x}_u \times \mathbf{x}_v\|} = \frac{|\boldsymbol{\omega} \cdot \mathbf{n}|}{r^2}. \tag{2.57}$$

# 3 Path Guiding with Vertex Triplet Distributions

As discussed in the previous chapter, good importance sampling strategies to keep variance low are crucial for the efficiency of Monte Carlo light transport simulation. However, designing sampling techniques that work well for a wide range of scenes is difficult, due to the strong dependency on visibility and materials. A promising approach is path guiding [VHH*19]: A path tracer augmented with guiding can often explore difficult regions of the path space, which would otherwise require bidirectional techniques, regularization (e.g. via vertex merging), or Markov chains. Guiding can thus often provide an unbiased, or more temporally stable alternative.

In this chapter, we present a novel path guiding algorithm. We observe that all terms in the measurement contribution function sampled during path construction depend on at most three consecutive path vertices. We thus propose to build a guiding distribution over vertex triplets that adapts to the full measurement contribution with a 9D Gaussian mixture model (GMM). This is able to express precise sampling distributions with bounded dimensionality. Since we model 3D vertex positions, our guiding distribution naturally extends to participating media. In addition, the symmetry in the GMM allows us to query it for paths constructed by both a path tracer and a light tracer from a jointly learned guiding distribution. Our method is able to learn and sample light transport paths with challenging interdependencies of path vertices introduced by glossy materials and participating media.

## 3.1 Introduction

Path guiding adapts path sampling techniques to the scene using information (e.g. incident illumination on surfaces) gathered from previously sampled paths. The gathering, representation, and exploitation of this information is decisive for the resulting path guiding algorithm: It has to be precise enough to capture small "islands" in the high-dimensional path space that contribute much energy to the image. At the same time, the representation needs to provide enough "fuzziness" to allow further exploration of the path space beyond discovered islands, but without extensive memory requirements.

Previous work has demonstrated path guiding, for example, based on the incident radiance at surface points [VKŠ*14; MGN17; RHL20], i.e. by approximating a 4D signal (two surface vertices, or vertex plus direction). This, however, is not enough information to account for the BSDFs when guiding paths at interactions. At the other extreme, Reibold et al. [RHJD18] store complete paths for guiding, but they can only sample similar paths and thus cannot share information about individual interactions between guide paths.

At the core of our work is the question: what is the simplest model that is as general as necessary for path guiding? It has to be at least as expressive as the physical equations defining the problem. Here

we observe that all terms in the measurement contribution function depend on a maximum of three consecutive path vertices. That is, (sets of) vertex triplets are able to represent all dependencies in path space such as visibility, or distances and phase functions in participating media. Furthermore, there is bidirectional symmetry in the transport equations: the adjoint operator is mostly the same due to reciprocity. We want to leverage this and use a single model trained by both light tracing and path tracing that can also be used to guide both directions.

Led by these requirements, we devise a 9D (three volume vertices) bidirectional model of the transport operator. The model is parametrized over incident vertex, scattering vertex, and outgoing vertex, and the quantity we are seeking to match is the measurement contribution function of the full path passing through this triplet. We use a Gaussian mixture model (GMM) to store this function, since it allows us to analytically compute conditionals and marginals as required during path sampling. To aid spatial stratification, we store such GMMs in cells of an octree. Each individual Gaussian component of these mixtures models the dependence of the measurement contribution on the input dimensions as linear correlation, which alleviates issues with interpolation between cells. In summary, we make the following contributions in this chapter:

- a general, 9D, bidirectional model of the transport operator including both surface and volume transport (Sections 3.3 and 3.4),
- bidirectional guided path construction using this model (Section 3.5),
- effective training of the model using reservoir-based resampling and a clustering approach based on principal component analysis (PCA) (Section 3.6).

## 3.2 Related work

### 3.2.1 Path guiding

Path guiding techniques use information from previously sampled light transport paths to construct new sampling densities which *guide* newly sampled paths to regions with high measurement contribution. They mostly differ in the model and data structure to represent this information, as well as the learning method to update approximations over time.

**Directional representations.** Most methods approximate the incident radiance field in the scene, which can then be used during forward path tracing to sample directions. Early works reconstruct hemispherical histograms of incident radiance for guiding paths either from photon maps [Jen95] or 5D trees [LW95]. Hey and Purgathofer [HP02] estimate the width of cones around important directions from a photon map. Vorba et al. [VKŠ*14] introduce 2D Gaussian mixture models (GMMs) that are fitted using online expectation maximization (EM). They train and render bidirectionally, but need to learn two separate unidirectional models instead of a unified bidirectional model like ours. Similar to Lafortune and Willems [LW95], Müller et al. [MGN17] introduce a 5D spatio-directional tree (SD-tree) to approximate incident radiance, which does not require expensive fitting and can be used for sampling directly. Zhu et al. [ZXS*21] use a neural network to combine light and camera path samples to

reconstruct an SD-tree for guiding. Bako et al. [BMDS19] train a neural network to reconstruct a local radiance field from neighboring samples and use it to guide importance sampling at the first bounce.

**High dimension.** Reibold et al. [RHJD18] take an opposite approach by identifying difficult paths and reconstructing a sampling density in the high-dimensional path space. While this helps explore particularly difficult and coherent regions in path space, their method suffers from the curse of dimensionality in less coherent regions. Similar problems occur in methods that learn in primary sample space [GBBE18; ZZ19]. In addition, primary sample space poses problems with recognizing coherent paths, as this information is quickly lost in higher dimensions.

**Spatial correlation.** While in the model of Reibold et al. [RHJD18] the 4D or 6D Gaussian distributions of consecutive path vertices account for the correlation of vertex positions, this information is not present in purely directional models. This can lead to problems when the far-field approximation of directional models breaks and spatial interpolation would be necessary. Müller et al. [Mül19] demonstrate how this can lead to artifacts on the borders of spatial regions of their SD-trees. They propose spatial filtering as a countermeasure, effectively blurring the directional representation. Ruppert et al. [RHL20] reproject directional mixture components using a mean depth value. While this can be accurate for diffuse emission at the hit point, it neglects visibility and angular constraints at glossy surfaces. Dodik et al. [DPÖM21] explicitly represent the correlations in a 5D spatio-directional mixture model. This is conceptually close to our approach, although they use incident radiance as a target function. The use of vertex positions in our model is another interesting difference: while the directional approach assumes a far-field incident radiance distribution and can deviate using correlation, our positional model starts with the parallax assumption and needs correlations to approximate parallel outgoing directions. Neural path guiding [MMR*19] learns a conditional model to sample the outgoing direction given an incident direction and position. Training and using this neural network for sampling is expensive and requires two GPUs to be competitive.

**Product guiding.** Local caches of incident radiance disregard the fact that the contribution of a full path also depends on the local BSDF at a vertex where the guiding cache is queried. Product path guiding [HEV*16] approaches this problem by combining an approximation of the BSDF with the incident radiance approximation during sampling and computing a product of GMMs. Similar approaches are adopted by other methods using mixture models [HZE*19; RHL20; DPÖM21]. Diolatzis et al. [DGJ*20] demonstrate a method to achieve product sampling with SD-trees.

One common challenge of product guiding is to find general models for representing complex materials. These need to be accurate and efficient to store, support textured parameters, and still allow efficient combination with the incident light representation. We take a different approach in directly approximating the full measurement contribution with our model.

**Volume guiding.** Path guiding in participating media is more challenging due to the necessity of distance sampling. Reibold et al. [RHJD18] directly sample 3D vertex positions or distances only from their 6D Gaussian representation. This is similar to our work, but still suffers from problems of high dimensionality. Distance sampling can also be achieved by stepping through a voxel data structure

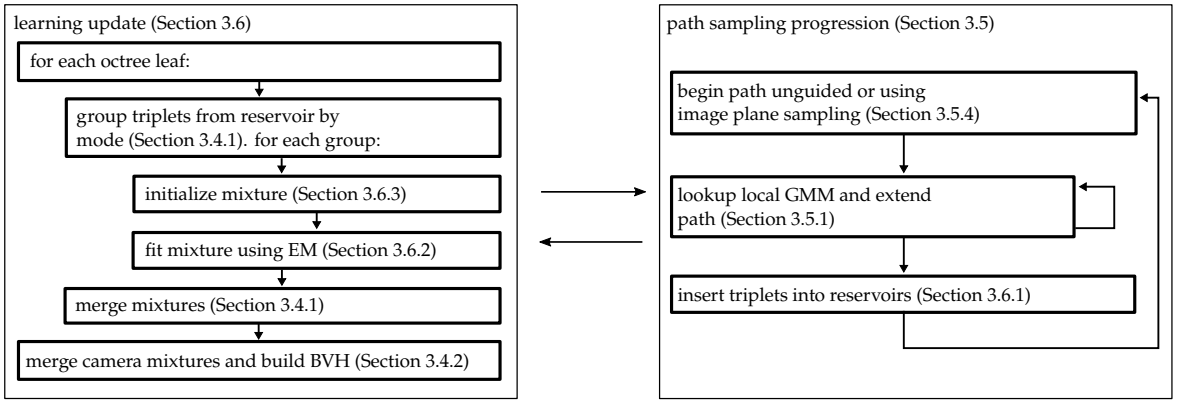



**Figure 3.1:** *Overview of the individual steps in our algorithm. During the learning phase, we alternate between learning (left) and path sampling (right).*

[HZE*19]. In contrast, our method can produce samples directly without incremental traversal of a data structure. Deng et al. [DWWH20] use SD-trees for directional guiding in media, but do not have a strategy for distance sampling.

### 3.2.2 Glossy light transport

Sampling chains of glossy interactions reliably is a hard light transport problem, as it is neither solved efficiently by bidirectional connection nor density estimation. Manifold walks [JM12; KHD14] have been proposed to efficiently explore these types of paths, leveraging the tridiagonal structure of the specular constraint derivative matrix. This structure is induced by the transport operator and also motivates our approach with vertex triplets. Unfortunately, manifold walks rely on derivatives that require locally smooth geometry. This assumption is broken by realistic scenes with high-frequency detail. While data-driven approaches cannot hope to achieve the same level of accuracy, they can act on global information, which is sometimes more relevant, for instance with highly detailed displacements.

## 3.3 Overview and theoretical background

The goal of our method is to learn a vertex triplet distribution that can be used to locally guide path sampling. We refer to this as the *guiding distribution* and model it using Gaussian mixture models (GMM). In this section, we provide an overview and a theoretical foundation of our model. We begin by justifying our choice of target function (Section 3.3.1), before defining the GMM and giving an intuition on how to sample it (Section 3.3.2). Further, we sketch how learning works and how it integrates into a renderer (Section 3.3.3).

In the subsequent sections, we discuss the complete algorithm: details on the guiding distribution (Section 3.4), its sampling (Section 3.5) and learning (Section 3.6). We show an overview of all the individual steps in Fig. 3.1.

### 3.3.1 Target function

We use the full measurement contribution function of a path as target function for the GMM. We will show that this results in a conditional PDF for the next path vertex which is proportional to the product of all terms remaining until completion of the path. In particular, part of this result is that the PDF does *not* depend on the terms that belong to the already sampled prefix.

First, we observe that the measurement contribution function can be factored into blocks $B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ that only depend on three consecutive vertices $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}$, i.e.

$$B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) T(\mathbf{x}_i, \mathbf{x}_{i+1}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \tag{3.1}$$

and we will use the shorthand $B_i = B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ indexed by the center vertex. Note that we arbitrarily separated the product such that $T$ and $G$ belong to one unique $B_i$. The complete measurement contribution function can then be written as

$$f(\overline{\mathbf{x}}) = W(\mathbf{x}_0) \cdot G(\mathbf{x}_0, \mathbf{x}_1) T(\mathbf{x}_0, \mathbf{x}_1) \prod_{i=1}^{k-2} B_i \cdot L_e(\mathbf{x}_{k-1}). \tag{3.2}$$

To extend a path in a unidirectional path tracer, we want to sample an outgoing position $\mathbf{x}_{i+1}$ given a path prefix that has already $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. The PDF $p(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{x}_{i-1})$ of $\mathbf{x}_{i+1}$ should be proportional to

$$p(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{x}_{i-1}) \sim B_i \cdot \int \prod_{j=i+1}^{k-2} B_j \cdot L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+2..k-1}. \tag{3.3}$$

Next, we will show that this is the case if the Gaussian mixture is adapted to the complete measurement contribution function. Conceptually, the GMM approximates a function $\mathcal{G}$ of three vertices $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}$. The model is fitted in a least squares sense against the full measurement contribution function $f$, only holding these three vertices fixed. Thus, this step integrates away all vertices $\mathbf{x}_j$ with $j < i-1$ or $j > i+1$:

$$\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \int f(\overline{\mathbf{x}}) \, \mathrm{d}\{\mathbf{x}_{j<i-1} \mathbf{x}_{j>i+1}\}. \tag{3.4}$$

Similarly, we can define

$$\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i) = \int \mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \, \mathrm{d}\mathbf{x}_{i+1} = \int f(\overline{\mathbf{x}}) \, \mathrm{d}\{\mathbf{x}_{j<i-1} \mathbf{x}_{j>i}\}, \tag{3.5}$$

which additionally integrates out all possible $\mathbf{x}_{i+1}$. This will give us, by definition of the conditional probability distribution:

$$\mathcal{G}(\mathbf{x}_{i+1}|\mathbf{x}_{i-1}, \mathbf{x}_i) = \frac{\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})}{\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i)}. \tag{3.6}$$

To analyze which terms of the measurement contribution function will stay in this expression and which will cancel out, we want to separate the integrals of products in Eqs. (3.4) and (3.5) into products of integrals. This is only possible if we can factor the integrand into separable parts. Unfortunately, the dependencies of the blocks $B_i$ on the vertices overlap, so in general this is not possible. For instance,

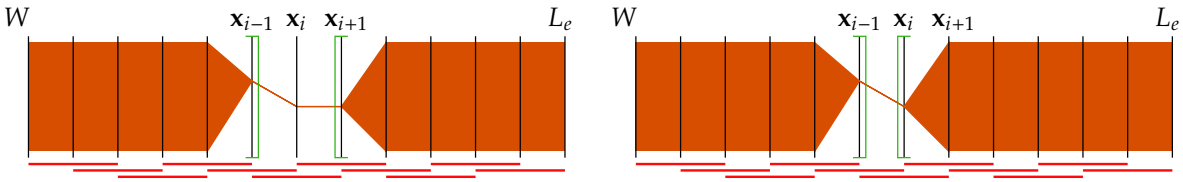



**Figure 3.2:** *Parallel coordinates visualization of path space. Separability of the path space integral (illustrated in orange) can be facilitated by holding consecutive vertices fixed. Since the BSDF $f_s$ depends on three consecutive vertices (indicated as red bars), we need to keep two fixed to be able to separate into two integrals. Left: numerator, right: denominator of the conditional in Eq.* (3.6)*. The green brackets show which terms can be separated, for instance $f_s(\mathbf{x}_{i-2}, \mathbf{x}_{i-1}, \mathbf{x}_i)$ is the last term in the left integral. Interestingly, the prefix over $\mathrm{d}\mathbf{x}_0 \cdots \mathbf{x}_{i-1}$ cancels out, but the postfix toward the light does not.*

consider the minimal example

$$I_1 = \int B_1 \cdot B_2 \cdot \mathrm{d}\mathbf{x}_{i=0..3} \tag{3.7}$$

which is not separable because both $B_i$ depend on $\mathbf{x}_1$ and $\mathbf{x}_2$. However, if we hold at least two consecutive vertices $\mathbf{x}_{i-1}, \mathbf{x}_i$ fixed, we can separate the integral:

$$I_2(\mathbf{x}_1, \mathbf{x}_2) = \int B_1 \cdot B_2 \mathrm{d}\mathbf{x}_0 \mathrm{d}\mathbf{x}_3 = \int B_1 \mathrm{d}\mathbf{x}_0 \cdot \int B_2 \mathrm{d}\mathbf{x}_3. \tag{3.8}$$

This allows us to separate both Eq. (3.4) and Eq. (3.5) in the numerator and denominator of Eq. (3.6) into two integrals. Figure 3.2 illustrates where this separation takes place: the prefix (the part of the path that has already been sampled and is held fixed) will cancel out in our conditional PDF, while the postfix (in this example the total flux through all connections to all light sources) stays:

$$\mathcal{G}(\mathbf{x}_{i+1}|\mathbf{x}_{i-1}, \mathbf{x}_i) = \frac{\cancel{\int WTG \prod_{j=1}^{i-1} B_j \mathrm{d}\mathbf{x}_{1..i-2}} \cdot B_i \cdot \int \prod_{j=i+1}^{k-2} B_j L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+2..k-1}}{\cancel{\int WTG \prod_{j=1}^{i-1} B_j \mathrm{d}\mathbf{x}_{1..i-2}} \cdot \int \prod_{j=i}^{k-2} B_j L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+1..k-1}} \tag{3.9}$$

The denominator contains a postfix integral as normalization factor: this is the same for all $\mathbf{x}_{i+1}$ we can possibly sample, since this vertex is integrated away. The rest is identical to the sought-for Eq. (3.3).

Note that we do not explicitly need to fit and store a GMM for Eq. (3.5), as we can take the conditional of the GMM that depends on three vertices in closed form. This will transparently yield the desired PDF in Eq. (3.3). This analysis is merely provided as proof for correctness as it might seem counterintuitive at first to use the full measurement contribution function as the target of a model that will be used to add vertices in unidirectional path tracing.

The symmetry of this result allows us to use this GMM for both training and sampling of path tracing and light tracing at the same time. This is a non-trivial result since when using partial throughputs of prefixes or suffixes, the global scale may be off by orders of magnitude. Previous work [VKŠ*14] therefore learns sampling for path and light tracing separately. The combination of guiding information from path and light tracing has only been achieved previously by using neural networks [ZXS*21].

### 3.3.2 Gaussian mixture model for path guiding

In our case, a vertex triplet $\mathbf{t} = (\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ is interpreted as the 9D vector of concatenated 3D vertex positions. We will assume $i = 2$ in this section to simplify notation, i.e. $\mathbf{t} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, but our equations also apply to the general case. To model the distribution of triplets, we use a 9D Gaussian mixture model with full covariance matrices, i.e. each of the $K$ components represents a weighted 9D normal distribution $\mathcal{N}(\mathbf{t} \mid \mu^k, \Sigma^k)$:

$$p(\mathbf{t}) = \sum_{k=1}^{K} \pi^k \, \mathcal{N}(\mathbf{t} \mid \mu^k, \Sigma^k), \tag{3.10}$$

$$\mu^k = (\mu_1^k, \mu_2^k, \mu_3^k) \in \mathbb{R}^9, \quad \Sigma^k = \begin{pmatrix} \Sigma_{11}^k & \Sigma_{12}^k & \Sigma_{13}^k \\ \Sigma_{21}^k & \Sigma_{22}^k & \Sigma_{23}^k \\ \Sigma_{31}^k & \Sigma_{32}^k & \Sigma_{33}^k \end{pmatrix} \in \mathbb{R}^{9\times 9}, \tag{3.11}$$

To simplify the notation, we will use $\pi = \pi^k$, $\mu = \mu^k$, $\Sigma = \Sigma^k$ whenever writing about single components.

**Guided sampling.** We use this GMM to locally guide sampling decisions when extending paths. Given a path prefix $(\ldots, \mathbf{x}_1, \mathbf{x}_2)$, we compute the conditional of this mixture given the last two vertices $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$\begin{aligned} & p(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2) \\ &= \sum_{k=1}^{K} \frac{\pi^k \, \mathcal{N}(\mathbf{x}_1, \mathbf{x}_2 \mid \mu_{1,2}^k, \Sigma_{1,2}^k)}{\sum_{l=1}^{K} \pi^l \, \mathcal{N}(\mathbf{x}_1, \mathbf{x}_2 \mid \mu_{1,2}^l, \Sigma_{1,2}^l)} \, \mathcal{N}(\mathbf{x}_3 \mid \mu_{3|1,2}^k, \Sigma_{3|1,2}^k) \\ &=: \sum_{k=1}^{K} \bar{\pi}^k \, \mathcal{N}(\mathbf{x}_3 \mid \bar{\mu}^k, \bar{\Sigma}^k). \end{aligned} \tag{3.12}$$

For the computation of the marginal distributions $\mathcal{N}(\mathbf{x}_1, \mathbf{x}_2 \mid \mu_{1,2}, \Sigma_{1,2})$ and conditional distributions $\mathcal{N}(\mathbf{x}_3 \mid \mu_{3|1,2}, \Sigma_{3|1,2})$ please refer to Appendix 3.A. This results in a 3D GMM which we sample in order to get the next vertex on the path. The details of this sampling procedure are presented in Section 3.5.

**Spatial data structure.** Learning a single global mixture model for an entire scene quickly turns infeasible. Similar to previous work, we instead partition the scene with an octree, where each leaf node contains a local GMM. For sampling and learning we then query the octree for the local GMM using the middle triplet vertex $\mathbf{x}_2$. We discuss the adaptation strategy for this octree in Section 3.6.1.

### 3.3.3 Algorithm

Our algorithm has two distinct phases: learning and rendering. We show an overview of all the individual steps in Fig. 3.1. In the learning phase, we iteratively sample from and update the guiding distribution after each progression. Once we reach the budget assigned to learning, we keep the guiding distribution fixed and start the rendering phase. Similar to most previous work, we discard the image

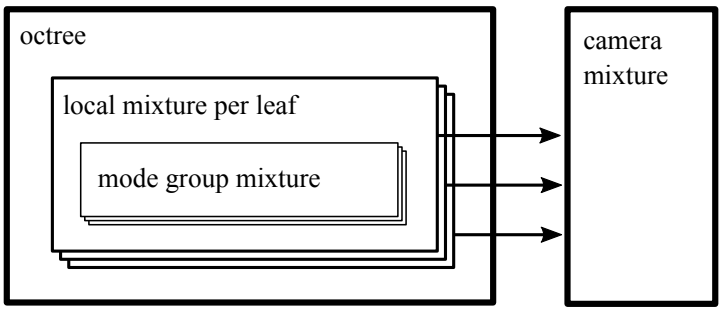



**Figure 3.3:** *In our guiding distribution, each leaf of the octree contains a local mixture used for sampling. These are constructed during learning by merging individually fitted mode group mixtures. The camera mixture is constructed for each leaf independently and then merged into the global one.*

rendered up to this point, as this prevents potential strong noise in the learning phase from affecting the final result. Alternatives to this have been conceived, notably discarding outlier noise via DBOR [ZHD18; RHJD18] or variance-weighted blending [Mül19]. We consider this orthogonal to our approach and leave this question outside the scope of our work.

During the learning phase, we resample triplets of completed paths online by storing them in reservoirs of fixed size local to each octree leaf. After each progression, we learn a new GMM for each octree leaf separately based on the triplets from the reservoir using expectation maximization (EM). We present details on the learning procedure, including the collection of triplets, initialization of EM and regularization of covariance matrices, in Section 3.6.

## 3.4 The guiding distribution

In practice, there are additional details about our guiding distribution beyond the GMM introduced so far. We will discuss these in the following, before moving on to sampling and learning. In particular, we partition triplets of different light transport modes (Section 3.4.1) and learn separate GMMs for each group. For sampling, we combine the different GMMs again, keeping transport mode data per component to inform sampling decisions. An important special case is a separate 3D GMM at the camera for sampling the image plane (Section 3.4.2). We show an overview of the different mixtures in our method in Fig. 3.3.

### 3.4.1 Light transport modes

In addition to the positions of each path vertex, we also have discrete metadata about the light transport attached to them. We refer to this as the light transport mode and distinguish between vertices in volumes, on surfaces, on the camera, and on environment maps. On surfaces, we distinguish between reflection and transmission events. Using this additional information for learning helps with separating multimodal distributions. For sampling, we use it for discrete decisions, e.g. to decide between sampling a volume or a surface vertex, or whether to reflect or transmit at a smooth dielectric surface (see Section 3.5.3).

To integrate this into our method, we store modes together with the vertex triplets. For learning, we

partition the triplets into groups with common combination of modes and build separate GMMs for each group. We combine them subsequently into a single GMM, storing the mode in each component. This becomes the local GMM stored in the octree leaf, which we will use for sampling. To keep the mixtures in relation, we need to scale the mixture component weights $\tilde{\pi}$ that were normalized to each individual mixture before. Consider $N$ triplets with triplet weights $w_i$, partitioned into groups defined by the index sets $I_1, \ldots, I_G$. For each mixture fitted to triplet group $g \in 1, \ldots, G$ we define its component weights as

$$\sum_{k=1}^{K_g} \tilde{\pi}_g^k = 1. \tag{3.13}$$

We combine the individual mixtures into a single mixture with $K = \sum_{g=1}^{G} K_g$ components and scale their weights proportionally to the sum of triplet weights from group $g$:

$$\pi_g^k = \tilde{\pi}_g^k \cdot \frac{\sum_{i \in I_g} w_i}{\sum_{i=1}^{N} w_i}. \tag{3.14}$$

This will allow us to treat this as a single GMM for sampling, where we just have to additionally compare light transport modes to select for valid components.

### 3.4.2 Mixture at the camera

Guided sampling of the first path vertex $\mathbf{x}_1$ after the camera can be important for allocating more samples to higher-variance areas in the image, exploring newly discovered features, and also for sampling distances in volumes. The latter is critical for sampling prominent volumetric effects well, e.g. light shafts. For this purpose, we maintain a camera GMM, which is only 3D and models the distribution of $\mathbf{x}_1$. When starting a new path, we will use it to sample a vertex position and connect to it from the camera. During the learning update, we build parts of the camera GMM separately for each octree leaf node using only the triplets originating from the camera. When all leaf camera mixtures are ready, we combine them into a global camera GMM, adjusting the component weights as described above for combining mixtures of different modes.

One additional difficulty with this approach is quickly evaluating the PDF of a large GMM for sampled paths. While for regular path extensions the number of components is limited, since we only need to consider the mixture in one octree leaf, the camera mixture can grow quite large. Therefore, we bound the support of components by using truncated Gaussians, as discussed in the following section. This allows us to compute axis-aligned bounding boxes (AABB) of camera mixture components and construct a 3D bounding volume hierarchy (BVH). We do so once per progression, after the learning update. Given a path vertex, a point query in the BVH now gives us all components that have this vertex in their support and can contribute to the PDF. This reduces the number of components we need to evaluate to a relatively small number. A similar approach for efficient PDF evaluation was used by Reibold et al. [RHJD18].

### 3.4.3 Truncated Gaussians

In place of true normal distributions, we always use truncated approximate Gaussians that are cut off at $4\sigma_j$ per dimension $j$, as described in the supplemental material of Reibold et al. [RHJD18]. The bounded support is particularly important for constructing the BVH for the camera mixture. For local mixtures, bounded support can also lead to many components evaluating to a zero conditional weight $\bar{\pi}$ early on. Thereby, we do not have to consider them for MIS, which overall improves efficiency. Unbounded support could potentially still be helpful, e.g. for cases where no mixture component has a non-zero weight.

## 3.5 Path sampling with vertex triplet distributions

In this section, we detail the construction of paths. On a high level, we follow this procedure: We construct paths starting from either the camera or a light source. Paths starting from the camera can be guided (Section 3.5.4), while we always start paths at the light sources with an unguided technique. Then we iteratively extend the path vertex by vertex, randomly selecting between guided or unguided sampling with a fixed probability $P_{\text{unguided}} = 0.5$. The combination with unguided sampling is necessary for exploration as well as unbiased results. For guided sampling, we conditionalize the GMM on the last two vertices of the current path prefix, and select a component from the conditionalized GMM (Section 3.5.1). Then we sample the position of, or direction to, the next vertex using the chosen component. Only in (near) specular cases (Section 3.5.3), we fall back to directional sampling of the BSDF or phase function. In addition, we perform next-event estimation at each vertex to efficiently handle cases that do not require guiding. For robustness, we combine all described sampling strategies with MIS using the balance heuristic.

### 3.5.1 Sampling the conditional GMM

To extend a path prefix $(\dots, \mathbf{x}_{i-1}, \mathbf{x}_i)$ by another vertex $\mathbf{x}_{i+1}$, we first look up the local 9D GMM from the octree based on $\mathbf{x}_i$, and conditionalize it on the last two vertices $\mathbf{x}_{i-1}, \mathbf{x}_i$ of the prefix as in Eq. (3.12):

$$p(\mathbf{x}_{i+1} \mid \mathbf{x}_{i-1}, \mathbf{x}_i) = \sum_{k=1}^{K} \bar{\pi}^k \, \mathcal{N}(\mathbf{x}_{i+1} \mid \bar{\mu}^k, \bar{\Sigma}^k). \tag{3.15}$$

In addition, we also disregard any components that cannot extend the current path because of contradicting light transport modes. Next, we randomly sample a component $k$ based on its conditional weight $\bar{\pi}^k$. The light transport mode of this component determines if we sample the next vertex on a surface or in a volume. For volumes, we directly sample the 3D position of the next vertex $\mathbf{x}_{i+1}$ using the conditional Gaussian $\mathcal{N}(\mathbf{x}_{i+1} \mid \bar{\mu}, \bar{\Sigma})$. For surfaces, we project the Gaussian to a plane first for easier PDF evaluation (Section 3.5.2). In both cases, as mentioned before, we fall back to directional sampling of BSDF or phase function for (near) specular materials at $\mathbf{x}_i$ (Section 3.5.3). After having sampled the new vertex, we check whether the light transport mode on the path matches that described by the mixture component and discard it otherwise. This happens e.g. when sampling a volume vertex that is occluded by surface geometry.

### 3.5.2 Sampling a surface vertex position

To sample a surface vertex, we could simply project a sampled 3D position to the scene geometry by tracing a ray. However, evaluating the PDF of the intersection point would require integration of the truncated Gaussian along the sampled ray, since any position on the ray could have produced the same intersection. To avoid this, we instead do the marginalization up front: we project the 3D Gaussian to a plane and sample the resulting 2D Gaussian. For any intersection of a ray with scene geometry, there is now only one corresponding point on the plane, for which we evaluate the PDF easily. As the projection plane, we use the one orthogonal to the connection of the current vertex $\mathbf{x}_i$ to the conditional mean $\bar{\mu}$. See Appendix 3.B for an alternative approach.

### 3.5.3 Handling (near) specular vertices

If the current vertex $\mathbf{x}_i$ is (near) specular, the outgoing direction is constrained and depends strongly on local geometry. Therefore, sampling a position $\mathbf{x}_{i+1}$ purely based on the conditionalized Gaussian $\mathcal{N}(\mathbf{x}_{i+1} \mid \bar{\mu}, \bar{\Sigma})$ will often be unfavorable or fail. Instead, we combine with BSDF or phase function sampling at $\mathbf{x}_i$ based on its roughness

$$r \coloneqq \begin{cases} \sqrt{\alpha} & \text{on microfacet surfaces with roughness } \alpha, \\ \sqrt{1 - |g|} & \text{in media with mean cosine } g. \end{cases} \tag{3.16}$$

by choosing BSDF or phase function sampling instead of position sampling with probability

$$P_{\text{BSDF}}(r) = \max\left(0, 1 - \frac{r}{r_{\max}}\right). \tag{3.17}$$

For roughness greater than $r_{\max} = 0.2$, we deem position sampling sufficient. Note that even so, we can still guide BSDF sampling with respect to discrete decisions, e.g. if we should reflect or transmit at a dielectric. Therefore, we consider this sampling technique a guided one, i.e. the probability for selecting it is

$$(1 - P_{\text{unguided}}) \cdot P_{\text{BSDF}}(r), \tag{3.18}$$

and thus the probability for guided position sampling is

$$(1 - P_{\text{unguided}}) \cdot (1 - P_{\text{BSDF}}(r)). \tag{3.19}$$

**Distance sampling.** When sampling a volume vertex, the BSDF or phase function only determines the direction. We can still leverage the conditional Gaussian for sampling a distance. For this, we transform the 3D covariance to a basis where one direction aligns with the sampled direction. Next, we conditionalize to the other two known dimensions (see Appendix 3.A) and obtain a conditional 1D Gaussian. Since we would need to reject negative distances, we sample only the positive part by remapping the random number using the CDF of the truncated Gaussian.

### 3.5.4 Image plane sampling

For each path starting from the camera, we choose between unguided or guided sampling of the image plane in the same way as when extending paths with probability $P_{\text{unguided}}$. To sample the vertex position $\mathbf{x}_1$, we proceed in the same way as when extending a path, using the fixed camera GMM (Section 3.4.2) in place of a conditional GMM. For MIS, we look up all components of the 3D GMM whose support includes the sampled vertex using the camera mixture BVH with a point query. To guarantee that this will in fact yield all components that could produce the sampled vertex, we terminate all paths where the sampled vertex projects outside of the AABB of the component support.

## 3.6 Learning a vertex triplet distribution with expectation maximization

Learning a guiding distribution involves multiple steps on which we provide detail in this section. Ultimately, we want to fit GMMs to vertex triplets using expectation maximization (EM) after each progression of the learning phase. As input, we require vertex triplets weighted according to our target function. We collect and resample triplets during path sampling in reservoirs. Keeping only a fixed number of triplets reduces the overhead of storing and processing many triplets later on. We review the reservoir data structure and its properties, and explain the weighting of triplets in Section 3.6.1. We review the EM algorithm for the case of GMMs in Section 3.6.2, where we also present our approach to regularizing the estimated covariances. To run EM, we further need an initial mixture that EM can improve upon subsequently. We detail our top-down clustering that we use as initialization in Section 3.6.3.

Triplet collection and fitting of mixtures are handled separately for each leaf node of the octree. We adapt the octree to the number of sampled triplets in each leaf, in order to achieve better approximation in important regions. As described in Section 3.4, in each leaf node we further partition vertex triplets into groups of different light transport modes for fitting, combining them only afterwards. Similarly, we construct the global camera mixture by fitting a part of it in each octree leaf separately. In the following, we will leave these details aside and focus on fitting single GMMs.

### 3.6.1 Collecting triplet samples

For each newly sampled path during learning, we split it into its vertex triplets. We assign each triplet to an octree cell, according to its middle vertex. Then, we insert the triplets into the reservoirs of the respective cells.

**Reservoir sampling.** Reservoir resampling naturally allows us to accumulate rare samples over multiple progressions, while keeping the total number of triplets in each octree cell fixed. We use the VarOpt [CDK*09] streaming resampling algorithm to achieve this. It maintains a reservoir of fixed size that consists of a subset $S \subset I$ of all previously inserted triplets. Triplets are inserted into the reservoir with a weight $w_i$, which is based on the target function. The probability for a triplet to be included in $S$ is proportional to the insertion weight $w_i$. When we take triplets out of the reservoir for EM, resampling

already partially accounted for these weights. As such, the output weights $\hat{w}_i$ of each triplet $i \in S$ will be different from their insertion weight $w_i$. Note that for discarded triplets $i \in I \setminus S$, the output weight $\hat{w}_i = 0$. In general, the output weights are unbiased estimates of the insertion weights:

$$\mathbb{E}[\hat{w}_i] = w_i, \tag{3.20}$$

and the sum of output weights equals the total sum of insertion weights

$$\sum_{i \in S} \hat{w}_i = \sum_{i \in I} w_i. \tag{3.21}$$

The reservoir data structure consists of an array of triplets with equal output weight and a priority queue with triplets with $\hat{w}_i = w_i$. If the insertion weight is too high to be included in the array by resampling, it is first inserted into the priority queue. When more triplets arrive later, this can lead to triplets being migrated from the priority queue to the equal-weight array. Hence, given a sufficient number of input samples, all output weights will become constant.

**Insertion weight of triplet samples.** Sampled paths $\bar{\mathbf{x}}$ follow a distribution that is different from the target we want to approximate with our guiding distribution. As shown in Section 3.3, our target density is proportional to the measurement contribution $f(\bar{\mathbf{x}})$. It follows that the normalized target density is

$$p_{\text{target}}(\bar{\mathbf{x}}) = \frac{1}{Z} f(\bar{\mathbf{x}}), \quad Z \coloneqq \int_{\mathcal{P}} f(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}. \tag{3.22}$$

Learning a model for the target distribution will amount to estimating the expectation of a function $s(\bar{\mathbf{x}})$ on paths

$$\mathbb{E}_{\text{target}}[s(\bar{\mathbf{x}})] = \int_{\mathcal{P}} s(\bar{\mathbf{x}}) \, p_{\text{target}}(\bar{\mathbf{x}}) \, \mathrm{d}\bar{\mathbf{x}}. \tag{3.23}$$

We can use importance sampling to express this expectation as one over the current path sampling density $p_{\text{current}}$:

$$\mathbb{E}_{\text{target}}[s(\bar{\mathbf{x}})] = \mathbb{E}_{\text{current}}\left[ s(\bar{\mathbf{x}}) \cdot \frac{p_{\text{target}}(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})} \right]. \tag{3.24}$$

Since the normalizing constant $Z$ is the unknown light transport integral, we cannot compute the importance weights for the unbiased estimator. However, we can still build on the self-normalized importance sampling estimator [Owe13, Ch. 9]:

$$\tilde{F}_n = \frac{\sum_{i=1}^{n} w(\bar{\mathbf{x}}_i) \, s(\bar{\mathbf{x}}_i)}{\sum_{i=1}^{n} w(\bar{\mathbf{x}}_i)}, \tag{3.25}$$

which is a consistent estimator for $\mathbb{E}_{\text{target}}[s(\bar{\mathbf{x}})]$ with the unnormalized importance weights

$$w(\bar{\mathbf{x}}) \coloneqq \frac{f(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})} = Z \cdot \frac{p_{\text{target}}(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})}. \tag{3.26}$$

Therefore, to insert triplets into reservoirs, we set their insertion weight to $w(\bar{\mathbf{x}})$.

**Spreading learned information.** As octree cells learn independently, each feature would have to be found by chance multiple times when it covers an area larger than the cell size. To spread information about discovered features, we stochastically put samples in neighboring cells. Similar to previous work [Mül19; RHL20], we randomly perturb the position used to look up the octree cell in a box with side length proportional (we use a factor of 0.2) to the current cell size. Note that we still keep the original triplet position for learning. For our method, this results in slightly larger mixture complexity, while for previous directional-only models this blurs the learned distribution over a larger area and is a trade-off with accuracy.

**Octree adaptation.** Like previous work, we adapt our spatial data structure to sample count. To keep the learning cost approximately fixed, we subdivide and collapse octree nodes to approximately reach a targeted number of leaf nodes $L$. We used $L = 300$ as a default value for our results. In each progression, we count how many triplets are inserted into the reservoirs of each octree leaf. Given the total triplet count $C$, we choose a threshold $T$:

$$T = \frac{C}{L}. \tag{3.27}$$

Before constructing mixtures in each leaf node, we collapse all nodes that received fewer triplets than $T$. Then we construct all mixtures and only afterwards subdivide octree cells that received more triplets than the threshold. We divide the triplets stored in reservoirs between the new leaf nodes based on their middle vertex positions. This results in the new leaf nodes using the same mixture for sampling, but being able to collect more triplets for the next progression. Importantly, it does not lead to mixtures being fitted to fewer triplets in newly subdivided leaf nodes, as would be the case if we subdivided before fitting.

### 3.6.2 Expectation maximization

After each progression, we rebuild the local GMMs in each octree leaf using expectation maximization (EM) [DLR77]. Given $N$ triplets with weights $w_i$ and positions $x_i \in \mathbb{R}^9$, we seek to maximize the log-likelihood function

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} w_i \log p(x_i|\theta), \tag{3.28}$$

where we abbreviated the GMM parameters as

$$\theta = (\pi^1, \mu^1, \Sigma^1, \ldots, \pi^K, \mu^K, \Sigma^K). \tag{3.29}$$

This is the same weighted formulation of the log-likelihood as used by Vorba et al. [VKŠ*14].

EM is an iterative approach to this problem by modeling assignment of triplets to components as unobserved variables. In the E-step of iteration $t$, we set the assignment of triplet $i$ to component $k$ to its expectation under current parameters $\theta_t$:

$$r_{ik} = \frac{\pi_t^k \mathcal{N}(x_i \mid \mu_t^k, \Sigma_t^k)}{\sum_{j=1}^{K} \pi_t^j \mathcal{N}(x_i \mid \mu_t^j, \Sigma_t^j)}. \tag{3.30}$$

Afterwards, in the M-step we maximize $\mathcal{L}(\theta_{t+1})$ given the weighted assignments $v_{ik} = w_i\, r_{ik}$:

$$\pi_{t+1}^k = \frac{\sum_{i=1}^N v_{ik}}{\sum_{i=1}^N w_i},\ \mu_{t+1}^k = \frac{\sum_{i=1}^N v_{ik}\, x_i}{\sum_{i=1}^N v_{ik}},\ \Sigma_{t+1}^k = \frac{\sum_{i=1}^N v_{ik}\ \left(x_i - \mu_{t+1}^k\right)\ \left(x_i - \mu_{t+1}^k\right)^{\mathrm{T}}}{\sum_{i=1}^N v_{ik}}. \tag{3.31}$$

Starting with an initial choice of parameters, we alternate both steps until a local maximum of $\mathcal{L}$ is reached. In practice, we stop once we reach a threshold ($\epsilon = 10^{-3}$) on the relative increase in log-likelihood:

$$\frac{|\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)|}{|\mathcal{L}(\theta_t)|} < \epsilon. \tag{3.32}$$

**Covariance regularization.** It is often the case that only few triplets are available for a mixture component, e.g. when a new light transport feature was randomly discovered. This makes it challenging to estimate high-dimensional covariance matrices. A common approach to regularize this problem is to estimate a structured covariance with fewer free parameters $\Sigma_{\mathrm{reg}}$ and use a convex combination of this with the sample covariance $\Sigma$ [Mur12, Ch. 4.6]:

$$\tilde{\Sigma} = \lambda\, \Sigma_{\mathrm{reg}} + (1 - \lambda)\, \Sigma. \tag{3.33}$$

When only very few samples are available, we fall back to a diagonal covariance with standard deviation $r$:

$$\Sigma_{\mathrm{iso}} = r^2\, I_9. \tag{3.34}$$

In our tests, we set $r = 0.1 \cdot c$ proportional to the octree cell size $c$. With more samples available, we found it helpful to switch to a data-dependent structure as soon as possible. For this structure, we could assume isotropic and independent distribution of vertex positions to end up with a diagonal covariance matrix and three variances to estimate. Since modeling vertex correlation is important to get accurate conditional distributions, we additionally assume isotropic correlation to arrive at

$$\Sigma_{\mathrm{block}} = \left(\frac{\mathrm{Tr}(\Sigma_{ij})}{3} \cdot I_3\right)_{i,j=1,2,3} \in \mathbb{R}^{9\times 9}, \tag{3.35}$$

and combine both into

$$\Sigma_{\mathrm{reg}} = \lambda_{\mathrm{block}}\, \Sigma_{\mathrm{block}} + (1 - \lambda_{\mathrm{block}})\, \Sigma_{\mathrm{iso}}. \tag{3.36}$$

We base the choices of $\lambda$ and $\lambda_{\mathrm{block}}$ on the sample size used in the covariance estimate. Since our samples are weighted, just counting them can severely underestimate correlation-like effects, e.g. when a single triplet with high weight dominates the estimate. Therefore, we use Kish's effective sample size estimator [Kis65] that accounts for this:

$$\hat{n} = \frac{\left(\sum_{i=1}^N v_i\right)^2}{\sum_{i=1}^N v_i^2}. \tag{3.37}$$

Then we set

$$\lambda = \frac{\hat{n} - 1}{(\hat{n} - 1) + 9}, \quad \lambda_{\mathrm{block}} = 1 - \min\left\{\frac{\hat{n} - 1}{3}, 1\right\}. \tag{3.38}$$

As an additional measure, we perform a singular value decomposition and clip the singular values to a

small positive value.

### 3.6.3 Initial mixture

The results of EM in general depend strongly on the initial parameters of the mixture $\theta_0$. In particular, the high dimensionality and strong correlations of triplet dimensions make this challenging. EM can often struggle to separate modes in the data given a bad initialization. Therefore, we invest effort into a top-down hard clustering of triplets before EM, which then typically leads to convergence within few iterations. This is one important reason for us to store triplets instead of accumulating statistics, as done in previous work [VKŠ*14; DPÖM21]. In contrast to Ruppert et al. [RHL20], we avoid evaluating merge criteria, but cannot make use of parameters from previous iterations in the MAP framework.

The idea of our clustering scheme is based on PCA-Part [SD07], which was originally intended for k-means. We briefly review PCA-Part and describe our modifications to make it more suitable for GMMs. These include improving cluster splits by EM iteration and a stopping condition for splitting clusters.

**PCA-Part.** The algorithm starts by putting all data points in a single cluster. It then splits this cluster in half along the direction of largest variation, creating two new clusters. The split direction is computed as the eigenvector of the covariance with the largest eigenvalue. This is repeated until the desired number of clusters is reached. The cluster with the worst fit to the data is picked as the next one to split. For k-means, the sum-squared-error is used to determine this score, but the authors already describe how to use the likelihood instead for the case of GMMs. Additionally, they discuss the possibility of trying all eigenvectors as possible split directions and use the split that results in the best score. We employ both of these modifications to the basic algorithm for our method.

**Improved cluster splits.** Splitting the triplets at planes oriented by eigenvectors and always at the mean can be too restrictive. Since the split effectively creates a two-component GMM, we can improve the assignment further by EM-iteration. These additional iterations can be amortized to a degree, because they typically lead to faster convergence for global EM later on. Also, they operate only on the subset of the triplets that were assigned to the cluster that is being split.

**Stopping condition.** To choose the number of components in our GMM, we set the maximum number to a high value (e.g. $K_{\max} = 128$) and evaluate a stopping condition before applying a cluster split. For this purpose, we compare the likelihood of the two-component GMM given by the split and the single Gaussian before splitting. To account for the different number of parameters $k$ in both models, we choose the model that is better according to its Akaike information criterion (AIC) [Aka73]

$$\mathrm{AIC}(k) = 2\,k - 2\,n\,\mathcal{L}(\theta(k)). \tag{3.39}$$

A similar approach was used in the x-means algorithm [PM00] in the context of k-means. They use the Bayesian information criterion (BIC) [Sch78] as a criterion, which we found to be too conservative for our application in practice. In this way, we will continue splitting clusters until each cluster is best represented by a single Gaussian.

## 3.7 Results

We implemented our method in a rendering system and provide source code online [SHJD22a].

### 3.7.1 Tested methods

We tested different variants of our method. We also implemented a restricted version that we refer to as *vertex pairs*. In this case, we just consider the pair of scattering and outgoing vertex. Compared to triplets, the incident vertex is not included in the model. This leads to a 6D GMM in place of the full 9D GMM, and this restricted version serves as a baseline to assess the benefit of using triplets. As this does not have the same symmetry, light tracing is not available for vertex pairs.

For vertex triplets, we make the use of light tracing optional. This is for comparison with unidirectional methods, and because it introduces overhead that does not necessarily amortize in all scenes. When light tracing is enabled, we always trace two paths for each pixel per progression: one camera path and one light path. In our results, we count this as two samples per pixel.
In total, we thus compare four different methods:

- Pairs: guided path tracing with vertex pairs,
- Triplets: guided path tracing with vertex triplets,
- Tripl+Llt: guided path tracing with vertex triplets, and guided light tracing during the learning phase only,
- Tripl+Flt: guided path tracing and light tracing with vertex triplets.

We compare our presented methods against the following state-of-the-art methods:

- Ppg: practical path guiding [MGN17], including improvements suggested later on [Mül19],
- Varpg: variance-aware path guiding [RGH*20],
- Vmm: parallax-aware path guiding [RHL20],
- Vmm+Prod: VMM with product importance sampling and guiding probability set to 0.25,
- Sdmm: path guiding using spatial-directional mixture models [DPÖM21],
- Zvvpg: zero-variance based volume path guiding [HZE*19],
- Spg: selective path guiding [RHJD18].

We use the default settings for parameters of all methods, unless otherwise noted.

**Normalized equal time comparison.** The different methods we compare are implemented in different rendering systems. To make equal-time comparisons fair, we normalize to the runtime of a path tracer with fixed sample count in each system. This approximately compensates for the overall overhead in these systems, but results should still be interpreted cautiously. We report the number of samples per pixel (spp) each method produced, and for methods with a distinct learning phase as learn+render spp.

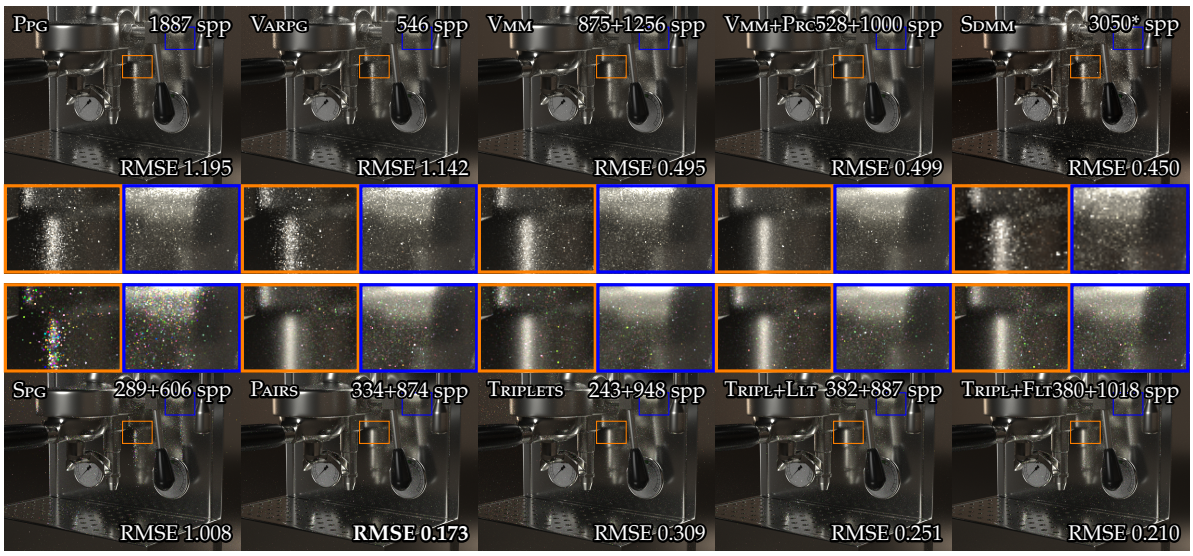



**Figure 3.4:** *Glossy surfaces lit by directional emitters produce hard-to-resolve reflections. Here, sampling the product of BSDF and incident radiance well is important, as can be seen in the difference between* Vmm *and* Vmm+Prod*, which achieves overall good results here.* Pairs *shows overall better performance than methods without product sampling.* Triplets *provide surprisingly little benefit over pairs. They struggle with learning the illumination on the background, although learning with the light tracer solves this problem. Note that the result of* Sdmm *is at half resolution, therefore the spp number is much higher than with the other methods.*

**Learning time allocation.** For algorithms that have a dedicated learning phase, we allocate 50% of the total time to learning. This includes the presented methods, Vmm and Spg. Ppg and Varpg have a similar learning schedule, due to their power-of-two iterations. For Zvvpg, we include the fixed light tracing and learning preprocess in the total time budget.

**Light tracing considerations.** As Zvvpg learns on light tracing paths, of our methods, Tripl+Llt is the most similar one in comparison. All other compared methods only make use of path tracing. This should be taken into consideration when comparing to our Tripl+Llt and especially Tripl+Flt results, but nevertheless the simple extension to guided light tracing is a benefit of our method. Note that when light tracing is enabled, we trace two paths per pixel per progression. This can result in a seemingly high number of spp, when many light paths terminate early.

**Color noise.** Our methods and Spg are implemented in a spectral rendering system, which produces colored noise that is not present in all other shown methods. Especially firefly samples are more noticeable in spectral renderings, as they tend to appear in bright colors.

### 3.7.2 Comparisons

Figure 3.4 shows difficult glossy surface transport, where sampling the product of BSDF and incident radiance is important. This should be a case where Triplets can show their full potential. Surprisingly, they show little benefit over Pairs, which already improves on methods without product sampling. This can be explained by our target function accounting for the full product, even for Pairs. Pairs is

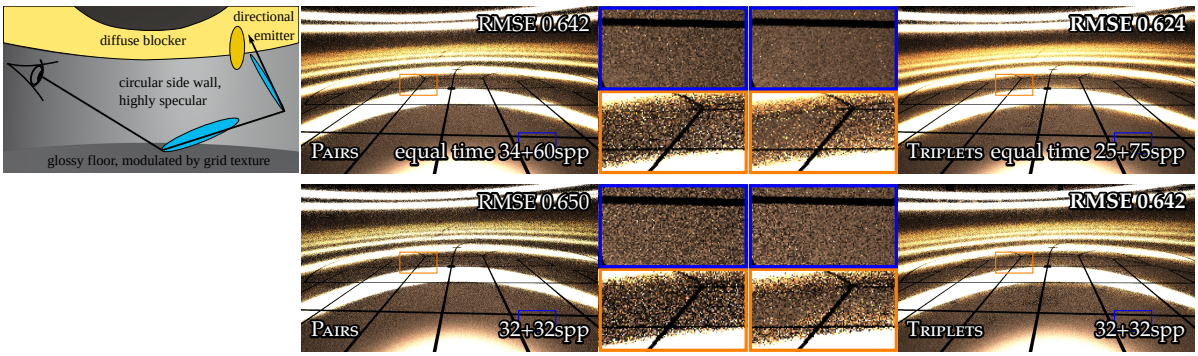



**Figure 3.5:** *A circular mirror chamber with a ring-shaped directional emitter on the ceiling. The ground floor has a glossy metal BRDF and is modulated by a grid texture for orientation. This simple scene shows the shortcomings of the GMM using pairs of vertices: it cannot distinguish the incident radiance field by outgoing direction. In this room, the rounded mirror shows spots on the ground texture from all sides: seen directly and seen in the mirror on all sides. This results in a suboptimal marginal distribution. The GMM using triplets can resolve this.* Triplets *also show some less converged areas, which we attribute to harder learning of the more general model.*

only missing the dependence on the incident vertex, but still encodes the product marginalized over all incident vertices. Often, this seems to be enough to be able to sample difficult paths at all, while the increased modeling power of Triplets cannot make up for harder learning. Note that for this comparison, we hit an error in the public implementation of SDMM. Therefore, we could only produce a result at half resolution.

Figure 3.5 shows a case where learning a model based on vertex pairs is not enough to resolve the reflectance field faithfully. We show equal sample and equal time comparisons between Pairs and Triplets. Even though the simpler Pairs approach can do roughly 35% more learning iterations in the same time, these do not pay off due to the simplistic model. Note that this performance gain is likely to shrink for more complex scenes with more significant cost of ray tracing and material evaluation.

In Fig. 3.6 we show a complex indirect volume caustic. Here, light tracing can help tremendously during both learning and rendering. However, the caustic seen in the mirror (orange inset) cannot be resolved by the light tracer, but works best for a guided path tracer. Thus, using the light tracing during learning only and then rendering using guided path tracing (Tripl+Llt) profits from good learning as well as fast iteration times during the rendering phase. Spg is very slow in this scene, likely due to higher-order scattering. Since a dedicated learning phase left only very few samples for rendering, we left the learning phase running for the whole render. This gave better results but still includes bright firefly samples.

Figure 3.7 shows a volume with anisotropic phase function in conjunction with glossy surface materials. Learning and rendering with Pairs is very fast, but fails to resolve the complex directional dependencies and cannot profit from paths constructed via light tracing. Zvvpg works well on the background fog but fails to capture the directed features caused by glossy reflections and peaky phase functions due to the lack of surface product sampling in this code path. Note, however, that the glow of the diffuse desert in the background is also not captured, even though it can be reproduced by careful distance and phase function sampling.

In Fig. 3.8 we show results on the pool scene, originally presented by Vorba et al. [VKŠ*14]. The

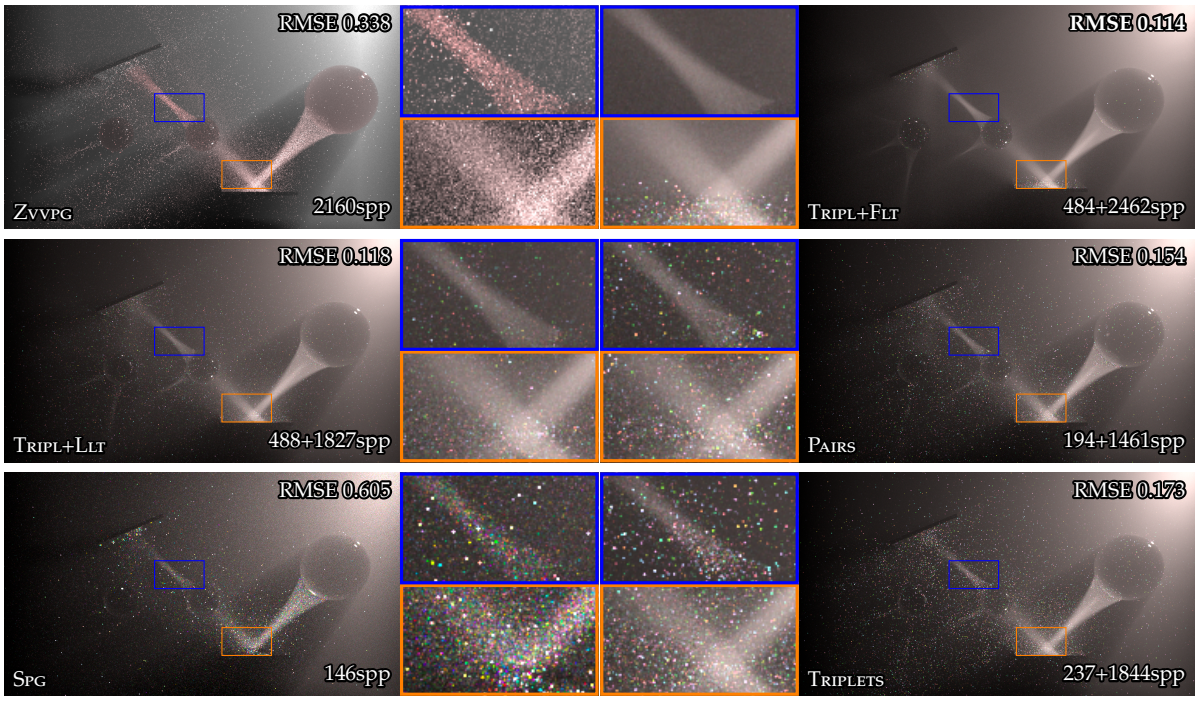



**Figure 3.6:** *Spheres and volume caustics by Herholz et al. [HZE*19]. Methods without light tracing (Pairs, Triplets, Spg) take a long time to learn dimmer caustics in the left part. Pairs is surprisingly slow in this scene. We hypothesize that this is due to our implementation not terminating long paths with little contribution using Russian roulette.*

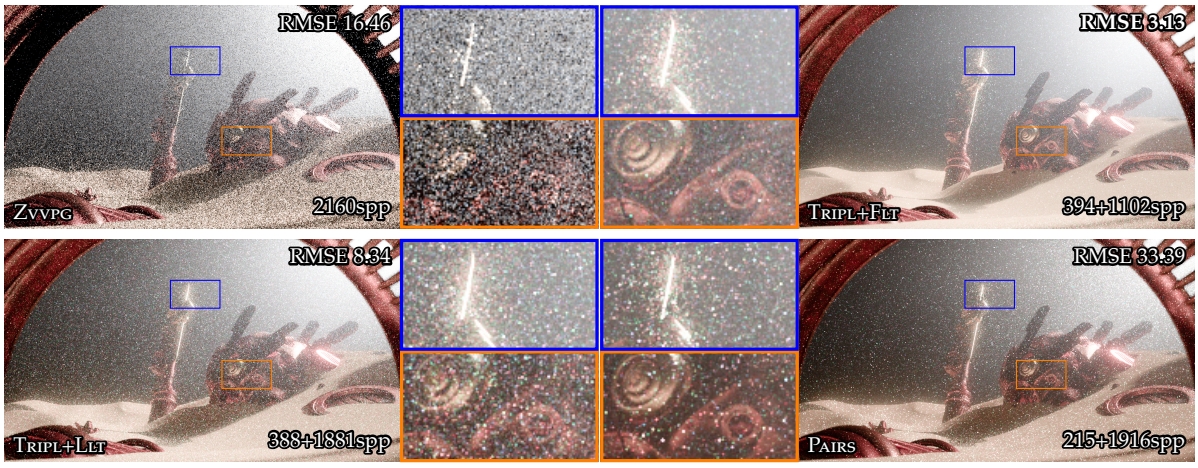



**Figure 3.7:** *A hazy scene lit by a directional emitter with mean cosine $g = 0.98$ in the homogeneous medium. Here, light tracing contributes important information to the learning as well as the rendering. Learning with Pairs is faster, so in this equal time comparison it achieves better results than learning with Triplets as long as light tracing is not used.*

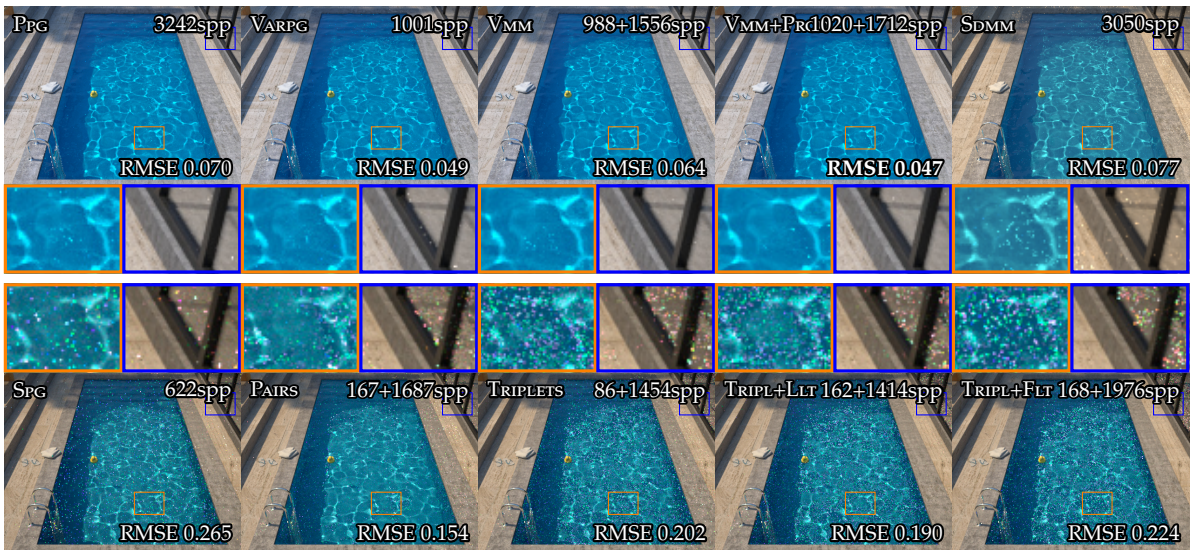



**Figure 3.8:** *The pool scene presents a simple case where for each 3D point only a 2D incoming radiance field has to be found. The bottom of the pool is diffuse, so there is nothing to be gained from a more complex approach and training a higher dimensional model just generates overhead both in terms of raw performance as well as convergence of the model itself.*

**Table 3.1:** *Our method uses a comparable amount of memory to previous work, shown here for the Pool (Fig. 3.8) and Espresso (Fig. 3.4) scenes. We separately list the memory requirements for storing the guiding distribution and for the samples that are used for learning.*

| | Vmm | Pairs | Triplets | Tripl+Flt |
|---|---|---|---|---|
| Espresso | | | | |
| guiding (MiB) | 41 | 9.5 | 18 | 33 |
| samples (MiB) | 184 | 106 | 140 | 144 |
| Pool | | | | |
| guiding (MiB) | 12 | 9.5 | 15 | 29 |
| samples (MiB) | 38 | 130 | 168 | 169 |

caustics on the pool floor in this scene are captured well by simpler models based on directional representation of incident radiance. Since the illumination comes from an infinitely far environment light, important directions are mostly the same for different positions on the pool floor. Our more complex model can still represent this, but requires more time for learning. Additionally, light tracing cannot connect to the camera through the dielectric water surface, and thus mostly does not help with learning.

We compare memory usage of our method in Table 3.1. Most of the memory is spent on storing triplet samples that are used for learning. Enabling light tracing increases memory requirements for the guiding distribution, because we precompute additional matrices. Overall, memory overhead is still low and of the same order as previous work.

## 3.8 Limitations and future work

We have shown how our model can represent and learn sampling densities for complex light transport scenarios. Further improvements can most likely be made by making learning more efficient, both in easy and hard cases.

**Selective guiding.** Many regions of path space are usually handled well by simple sampling strategies. Currently, our method does not exploit this fact and needs to handle the full path space. It would be an interesting extension to selectively focus on difficult regions, like in the method of Reibold et al. [RHJD18]. One difficulty with applying their approach is obtaining a stable target function: DBOR is always relative to the current path sampling density, which is changing during learning.

Another interesting aspect for future work would be to determine the "difficulty" of individual triplets, as not all parts of the path are usually equally hard to sample. Our model can use this information when available, and e.g. select the guiding probability for each component differently, thus taking position and incident direction into account for this decision.

**Variance-awareness.** Our target function is unaware of variance in the estimators. Rath et al. [RGH*20] show how the square root of the second moment should be the target for minimizing variance. However, their approach does not easily transfer to mixture models trained using EM. Further research on how their results can be incorporated into more complex models could help with better distribution of samples to high-variance regions. This would be especially interesting for quick exploration of these regions during learning.

**Path space regularization.** As our method learns from full transport paths, we cannot make use of partially constructed paths, e.g. when a light tracer cannot connect to the camera through a specular interface. A possible solution is to form connections (during learning) using path space regularization [KD13]. Previous work showed how this can benefit path guiding [WDH*21] by discovering features more easily.

## 3.9 Conclusion

We introduced a general data model to represent the light transport operator in image synthesis. This model was crafted to be as general as possibly needed to express all dependencies arising between path vertex geometry and the transported differential flux, but with no redundancy. The result is a 9D mixture model that depends on three path vertices: incoming, scattering, and outgoing. Furthermore, it is split into components for scatter modes such as reflect or transmit. We can transparently train a single model and sample from it using bidirectionally constructed paths, which was previously only done using neural networks. We demonstrated that we can learn and sample intricate directional dependencies for glossy surface scattering as well as volumetric transport including distance sampling.

As a general technique, the biggest future challenge is to reduce the overhead over simpler approaches that are sufficient for scenes with simpler configurations. For instance, the pool scene has a diffuse material under the caustic and can be well represented by a lower-dimensional model that is able to

learn and render faster. We expect that future improvements to our model are possible, especially in the learning phase as discussed in this chapter.

## 3.A Conditional and marginal normal distributions

Given normal distributed $(x_1, x_2) \sim \mathcal{N}(\mu, \Sigma)$ with (block) covariance and mean

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \tag{3.40}$$

the marginal distribution of $x_2$ is simply

$$x_2 \sim \mathcal{N}(\mu_2, \Sigma_{22}). \tag{3.41}$$

When $x_1$ is known, we can compute the conditional distribution of $x_2$

$$x_{2|1} \sim \mathcal{N}(\mu_{2|1}, \Sigma_{2|1}) \tag{3.42}$$

using

$$\begin{aligned} \mu_{2|1} &= \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \\ \Sigma_{2|1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \end{aligned} \tag{3.43}$$

Here, $\Sigma_{11}^{-1}$ denotes the generalized inverse in case $\Sigma_{11}$ is singular.

## 3.B Exact sampling of 3D Gaussians for surface vertices

To sample a point $\mathbf{y}$ on a surface using a 3D Gaussian $\mathcal{N}(\mu_0, \Sigma)$, we first sample a point in space $\mathbf{y}'$ and then project it to the scene geometry $\mathbf{x}$. Effectively, we are sampling the direction $\omega = \frac{\mathbf{y}'-\mathbf{x}}{\|\mathbf{y}'-\mathbf{x}\|}$, where $\mathbf{x}$ is the current vertex position. In the estimator, we need the solid angle PDF of $\omega$. If we sampled $\mathbf{y}'$ directly using its 3D distribution, we would need to marginalize out the additional dimension to get the 2D solid angle PDF:

$$p(\omega) = \int_0^\infty \mathcal{N}(\mathbf{x} + r\omega \mid \mu_0, \Sigma)\, r^2 \,\mathrm{d}r. \tag{3.44}$$

Therefore, we use an approximation in Section 3.5.2, where we sample $\mathbf{y}'$ using a 2D distribution. We first project the 3D Gaussian to a plane orthogonal to $\mu_0 - \mathbf{x}$ and then sample the resulting 2D Gaussian, allowing us to compute the solid angle PDF easily.

In a follow-up investigation, we realized that we could compute the value of the marginal integral Eq. (3.44) efficiently, making exact sampling of the 3D Gaussian practical. For this, we sample $\mathbf{y}' \sim \mathcal{N}(\mu_0, \Sigma)$, resulting in the direction $\omega$. To compute the PDF, we rewrite the marginal integral (Eq. (3.44)) using $\mu \coloneqq \mu_0 - \mathbf{x}$, plug in the 3D Gaussian PDF and use a computer algebra system to solve

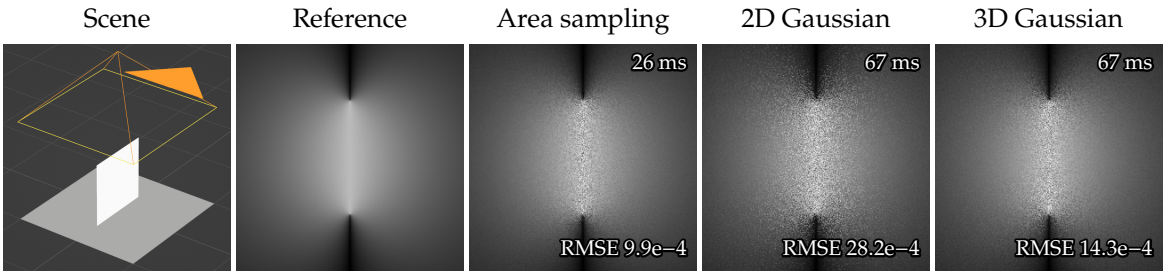



**Figure 3.9:** *A simple scene with a square light illuminating a plane (left). Uniform sampling of points on the square (area sampling) is only slightly better than sampling its 3D Gaussian representation. While accurate in the far field, projecting to a 2D Gaussian first leads to increased noise near the light.*

the remaining integral:

$$p(\omega) = \int_0^\infty \mathcal{N}(\mathbf{x} + r\omega \mid \mu_0, \Sigma)\, r^2 \,\mathrm{d}r = \int_0^\infty \mathcal{N}(r\omega \mid \mu, \Sigma)\, r^2 \,\mathrm{d}r \tag{3.45a}$$

$$= \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma|}} \int_0^\infty \exp\left(-\frac{1}{2}(r\omega - \mu)^\mathrm{T}\Sigma^{-1}(r\omega - \mu)\right) r^2 \,\mathrm{d}r \tag{3.45b}$$

$$= \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma|}} \int_0^\infty \exp\left(-\frac{1}{2}\Big(r^2 \underbrace{\omega^\mathrm{T}\Sigma^{-1}\omega}_{=:a} - 2r \underbrace{\omega^\mathrm{T}\Sigma^{-1}\mu}_{=:b} + \underbrace{\mu^\mathrm{T}\Sigma^{-1}\mu}_{=:c}\Big)\right) r^2 \,\mathrm{d}r \tag{3.45c}$$

$$= \frac{1}{(2\pi)^{\frac{3}{2}}\sqrt{|\Sigma|}} \int_0^\infty \exp\left(-\frac{a}{2}r^2 + br - \frac{c}{2}\right) r^2 \,\mathrm{d}r \tag{3.45d}$$

$$= \frac{1}{(2a)^{\frac{5}{2}}\pi^{\frac{3}{2}}\sqrt{|\Sigma|}} \left[\sqrt{2\pi}\,(a + b^2)\left(\operatorname{erf}\left(\frac{b}{\sqrt{2a}}\right) + 1\right)\exp\left(\frac{b^2}{2a} - \frac{c}{2}\right) + 2b\sqrt{a}\,\exp\left(-\frac{c}{2}\right)\right] \tag{3.45e}$$

$$= \frac{1}{(2a)^{\frac{5}{2}}\pi^{\frac{3}{2}}\sqrt{|\Sigma|}} \left[\sqrt{2\pi}\,(a + b^2)\left(\operatorname{erf}\left(\frac{b}{\sqrt{2a}}\right) + 1\right)\exp\left(-\frac{d}{2a|\Sigma|}\right) + 2b\sqrt{a}\,\exp\left(-\frac{c}{2}\right)\right]. \tag{3.45f}$$

In the last step, we sidestep catastrophic cancellation in the exponent by substituting using

$$\underbrace{\omega^\mathrm{T}\Sigma^{-1}\omega}_{=a}\,\underbrace{\mu^\mathrm{T}\Sigma^{-1}\mu}_{=c} - \underbrace{\left(\omega^\mathrm{T}\Sigma^{-1}\mu\right)^2}_{=b} = \frac{1}{|\Sigma|}\underbrace{(\omega \times \mu)^\mathrm{T}\Sigma(\omega \times \mu)}_{=:d}. \tag{3.46}$$

In Fig. 3.9, we compare this sampling technique (3D Gaussian) to our previous one (2D Gaussian) in a simple test scene. We sample points using a Gaussian centered on the unit square light source with $\Sigma = \operatorname{diag}\left(1\mathrm{e}{-4}, \frac{1}{12}, \frac{1}{12}\right)$. For each sampling technique we use a total of 128 samples per pixel, while ensuring domain coverage using MIS combination with 10% cosine-weighted hemisphere samples. Sampling of the 3D Gaussian is more accurate in the near field than the previous 2D Gaussian sampling. As both methods are equally fast in this simple test case, evaluating the long PDF expression (Eq. (3.45f)) does not seem to introduce significant overhead. Since there is no obvious disadvantage to the sampling technique from Section 3.5.2, it makes sense to use the 3D sampling technique with the above PDF as a substitute, even if better near-field accuracy might not be as significant in practical application.

# 4 Bridge Sampling for Connections via Multiple Scattering Events

To handle specific importance sampling problems, specialized solutions can complement more general approaches toward sampling difficult light transport such as path guiding. For example, explicit connections to light sources leverage prior knowledge about the distribution of illumination in a scene. Using connections for next-event estimation improves exploration of path guiding and allows the guiding distribution to focus on the indirect illumination [HSRM25]. In dense, forward-scattering participating media, however, the benefit of connections declines, as significant transport happens over longer multiple-scattering paths around the straight connection to the light.

In this chapter, we present a method to recover the benefit of connections by importance sampling a *bridge*: a subpath of arbitrary vertex count connecting two vertices. This is challenging, as the contribution of a bridge is shaped by the product of phase functions. Therefore, we importance sample the phase functions first, and subsequently construct a geometrically valid bridge. This approach leads to a simple algorithm and can construct bridges of any vertex count. However, the probability density of the sampled bridge is nontrivial. By formulating it as a marginal density and solving the resulting integral, we obtain an analytical expression. We demonstrate the effectiveness of our method in reducing estimator variance in various scene configurations with forward-scattering media.

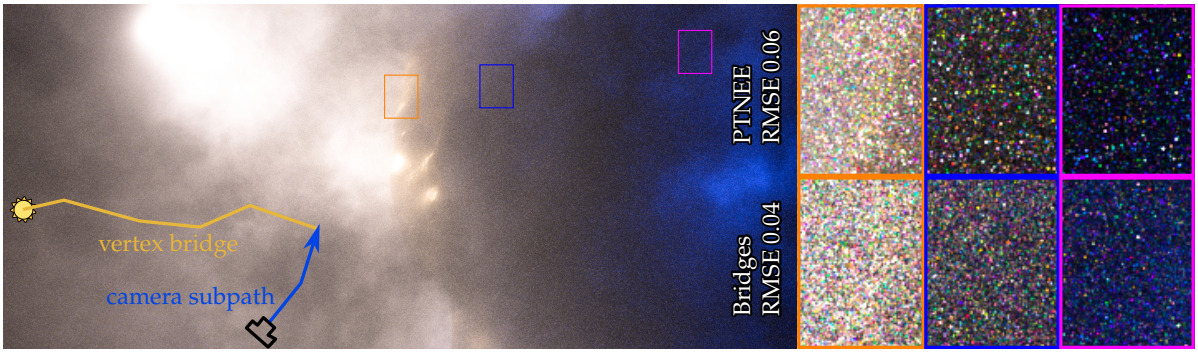



**Figure 4.1:** *A golden statue submerged in a heterogeneous cloud volume (scattering albedo 100%, 404M 4D voxels with motion blur, the phase function approximates Mie scattering by an equal mixture of two Henyey-Greenstein lobes of mean cosines $g = 0.98$ and $g = -0.4$). The image is rendered up to a maximum path length of 10 vertices. We importance sample the multiple scattering contribution by* bridges *which take into account the phase functions and the geometry terms of the measurement contribution. In an approximate equal time comparison, this leads to much improved rendition of the multiple scattering (blue/magenta insets where the statue shadows the indirect illumination in the volume). PTNEE resolves the direct illumination better (orange inset near the white light), though neglecting the multiple scattering.*

## 4.1 Introduction

Common participating media, e.g. fog or water, exhibit high albedo and highly peaked forward scattering. Especially for importance sampling the light emission, explicit connections of subpaths are usually significant, but sampling straight line connections is less effective in these media. This is because a major share of the light transport happens via sequences of multiple scattering events. These form paths around the straight connection, as shown in the example in Fig. 4.2. It also demonstrates how path tracing with next-event estimation struggles to sample these paths. Like all methods based on analog path extensions, it tends to get lost in less important regions of the medium.

Our goal is to explicitly sample this type of connecting subpath between two vertices, which we refer to as a *bridge*. This is challenging, since we need to handle two aspects of the geometric configuration along the bridge: the forward-scattering peak in the phase functions leads to near-specular transport, and the geometric singularity in the distance between vertices can cause infinite variance. Therefore, we need to importance sample the product of these terms for robust sampling.

Fortunately, we can turn to an observation about analog path tracing: it is mostly very good at sampling paths, with the critical flaw of terminating at a random point instead of our target vertex. Our main idea is to fix this *after* sampling a preliminary path: we rotate and uniformly scale it, such that its endpoint will match the target. By preserving angles, this transformation maintains the important geometric properties of the initial sampling. The probability density of this sampling technique involves marginalization, i.e. for a given bridge we need to integrate over all ways in which we could have sampled it. This turns out to be surprisingly simple for our purposes and we derive an analytical result.

Sampling this way, we forego importance sampling of the directions at endpoints, though other vertex connection techniques share this property. This is usually handled well by combination through multiple importance sampling. We also cannot control the total track length and thus transmittance, but typically this does not cause a lot of variance in the case of forward scattering. All other terms of the bridge are importance sampled by our algorithm.
In summary, we make the following contributions:

- a general formulation of the bridge sampling algorithm and its PDF derivation (Section 4.3.1),
- our recommended instance of this algorithm, which samples distances on bridges of arbitrary vertex count based on scaling the bridge (Section 4.3.2),
- a second instance without scaling or marginalization, which works with one or two inserted vertices (Section 4.3.3),
- an algorithm for sampling the number of vertices on a bridge, based on connection distance and medium properties (Section 4.4).

## 4.2 Related work

**Extended next-event estimation in volumes.** Rendering participating media is hard, especially in the presence of high-frequency content in the phase functions (most commonly: very peaked forward scattering, but Mie theory also predicts backward peaks) since this poses angular constraints on

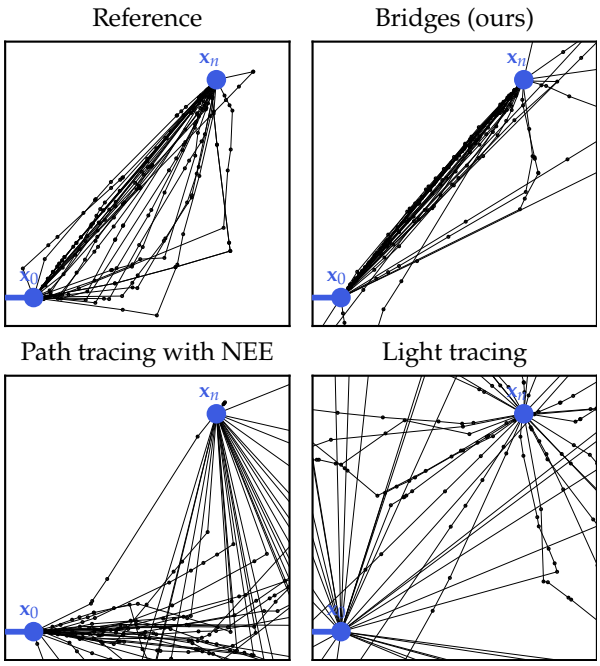



**Figure 4.2:** *Path distributions produced by different sampling algorithms for connecting a subpath ending at* $\mathbf{x}_0$ *to a light source at* $\mathbf{x}_n$ *over* 8 *additional vertices in an infinite medium with Henyey-Greenstein phase function and mean cosine* $g = 0.95$*. The random walk of both path and light tracing is uninformed about the goal vertex and struggles to focus on the important regions of path space. Bridge sampling is conditional on both endpoints and successfully samples paths near the straight line connection. The reference is the distribution of paths that would be optimal for importance sampling, i.e. proportional to their measurement contribution. It also shows paths that first follow the forward-scattering phase function to the right, before turning toward the light source, which in a complete renderer would be handled by MIS combination of our sampling and phase function sampling. We compute the reference by resampling 30 paths from 6000000 candidates generated by path tracing with NEE.*

paths with high throughput. Even without this, the geometry terms $G$ contained in the measurement contribution are a challenge. The inverse square distance term is unbounded if path vertices come closer together. If light sources are contained in a volume, this term can cause infinite variance in the estimator. To counter this, equiangular sampling includes the offending $G$ term in the importance sampling [KF12]. This method works best for isotropic phase functions, since these are omitted from the PDF. To address this, Georgiev et al. [GKH*13] propose joint importance sampling of distances and phase functions, but resort to tabulation, which still results in suboptimal sampling for generic and spiky phase functions. *Once More Scattered NEE* (OMNEE) solves the special case of a single additional scattering vertex between a vertex and the light source for homogeneous media and arbitrary forward scattering phase functions [HWD22]. Our method works for full-sphere phase functions and can insert an arbitrary number of extra vertices, and we show an equivalent formulation for the special case of a single vertex. *Multiple vertex next-event estimation* (MVNEE) [WHD17] adds multiple vertices on the way to the light, too. As pointed out by the authors and confirmed by our results, the resulting PDF is not ideal, especially for longer paths.

**Glossy light transport.** Solving for a path with multiple vertices that fulfills angular constraints dictated by the phase functions is similar in spirit to highly glossy/specular light transport on surfaces.

This has been explored for Markov chain rendering [Jak13] as well as for extending Monte Carlo next-event estimation [HDF15]. Our formulation is much simpler and works without an iterative predictor/corrector scheme since path vertices in the medium are less constrained as compared to surfaces.

**Product sampling.** Sampling the product of material properties and incident light [HEV*16] has been explored for volumetric scattering [HZE*19], by casting both incident light and phase function to a Gaussian mixture model and subsequently computing the product of all mixed-term Gaussians in closed form. As shown previously, the product of incident light and a peaky phase function can become bi-modal [HWD22]. This means a product importance sampling scheme cannot simply rely on the product of two Gaussian approximations, which would remain unimodal. Thus, such an approach will require many lobes in a Gaussian mixture to obtain good accuracy.

**Caching light transport.** Employing memory to store light paths to compute volumetric light transport has been explored in great detail [KGH*14]. This incurs an implementation, memory, and runtime overhead to evaluate the PDF of all active estimators to compute the multiple importance sampling weights. Path guiding [VHH*19] employs memory to explicitly store sampling densities to guide random walks toward high-contribution paths. This approach also handles visibility, while our importance sampling scheme does not. A specialized class of algorithms is concerned with subsurface scattering, and guides transport paths back to the surface [Kd14].

**Approximations for high-order scattering.** Analytic expressions for multiple scattering are hard to obtain and represent a well-researched area [dEo16]. These formulations are especially useful for extremely long paths and when the visibility between transport vertices is known to be unobstructed or can be ignored. With our method, we address a scattering regime up to the lower two-digit vertex counts.

**Time-of-flight imaging.** Connections via additional scattering vertices in media [JMM*14] or on surfaces [PVG19] can also be used to control the total track lengths of light paths. This is important when considering the time-of-flight of light, while we focus on the more common steady-state simulation. Our sampling aims to fulfill angular constraints on multiple additional scattering vertices in media, while track length is only of secondary concern.

## 4.3 Sampling vertex bridges

Our goal is to connect two vertices $\mathbf{x}_0$ and $\mathbf{x}_n$ via a chain of $n$ edges, while importance sampling all inner phase functions and reciprocal squared distance terms. To accomplish this, we first sample a preliminary subpath $\bar{\mathbf{x}}'_{0,n}$ starting from $\mathbf{x}'_0 = \mathbf{x}_0$, which we call a *bridge*. Next, we rotate and scale the bridge around $\mathbf{x}_0$, such that its endpoint $\mathbf{x}'_n$ matches the target $\mathbf{x}_n$, as shown in Fig. 4.3. This affine transformation preserves angles between edges and therefore the phase function PDF under mild assumptions (see Section 4.3.1). Therefore, we can easily make use of stock phase function sampling while constructing

**Table 4.1:** *Summary of our notation.*

| Term | Description |
|---|---|
| $\mathbf{x} \in X,\ \mathbf{x}' \in X'$ | (Preliminary) path vertex position |
| $\boldsymbol{\omega} \in \Omega,\ \boldsymbol{\omega}' \in \Omega'$ | (Preliminary) path edge direction |
| $d \in D,\ d' \in D'$ | (Preliminary) path vertex distance |
| $\boldsymbol{\omega}_{a\to b},\ \boldsymbol{\omega}'_{a\to b}$ | direction from $\mathbf{x}_a$ to $\mathbf{x}_b$ ($\mathbf{x}'_a$ to $\mathbf{x}'_b$) |
| $\omega_i,\ \omega'_i$ | direction from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ ($\mathbf{x}'_{i-1}$ to $\mathbf{x}'_i$) |
| $d_i,\ d'_i$ | distance between $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ ($\mathbf{x}'_{i-1}$ and $\mathbf{x}'_i$) |
| $\bar{\mathbf{x}}_{i,j} \in \bar{X}$ | the sequence $(\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j)$ |
| $\bar{\boldsymbol{\omega}}_{i,j} \in \bar{\Omega}$ | the sequence $(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i+1}, \ldots, \boldsymbol{\omega}_j)$ |
| $\bar{d}_{i,j} \in \bar{D}$ | the sequence $(d_i, d_{i+1}, \ldots, d_j)$ |
| $\bar{\mathbf{x}},\ \bar{\boldsymbol{\omega}},\ \bar{d}$ | shorthand for $\bar{\mathbf{x}}_{0,n},\ \bar{\boldsymbol{\omega}}_{0,n},\ \bar{d}_{0,n}$ |
| $\bar{\mathbf{x}}',\ \bar{\boldsymbol{\omega}}',\ \bar{d}'$ | analogous to above |

the bridge. Since we stay in the solid angle domain, all reciprocal squared distance terms are sampled naturally as well. Please see Table 4.1 for a description of our notation.

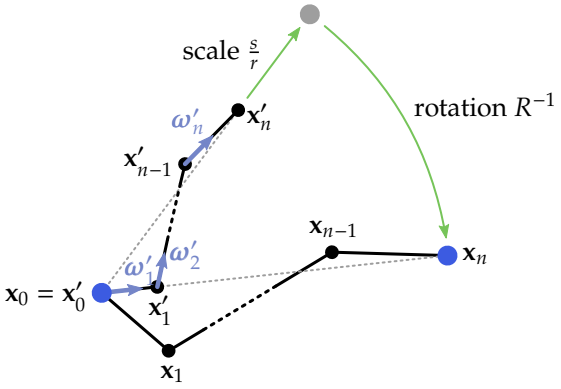



**Figure 4.3:** *Our goal is to connect the endpoints $\mathbf{x}_0$ and $\mathbf{x}_n$ via a subpath with $n-1$ additional vertices and importance-sampled directions $\bar{\boldsymbol{\omega}}_{2,n} = (\boldsymbol{\omega}'_2, \ldots, \boldsymbol{\omega}'_n)$. We sample a preliminary path $\bar{\mathbf{x}}'_{0,n}$ beginning from $\mathbf{x}'_0 = \mathbf{x}_0$ and $\boldsymbol{\omega}'_1 = \boldsymbol{\omega}_{0\to n}$, and then proceed via free-flight and phase function sampling. To connect $\mathbf{x}_0$ and $\mathbf{x}_n$, we rotate and scale this path around $\mathbf{x}_0$ such that $\mathbf{x}'_n$ ends up at $\mathbf{x}_n$. Notably, this affine transformation preserves angles and thus phase function importance sampling weights.*

The remaining challenge is to sample suitable vertex distances and compute the PDF of the resulting vertices. In this section, we present two alternatives for distance sampling. The first one samples preliminary independent distances using the exponential distribution and realizes the correct bridge length through scaling (Section 4.3.2). It is simple to implement for any number of vertices, but incurs a non-constant Jacobian for its projection step. The second alternative can get rid of this factor by directly sampling solutions to the constrained problem (Section 4.3.3), i.e. it always samples a unit scale factor, but is limited to inserting a maximum of two vertices. Beforehand, we will derive the PDF of our sampling scheme, independently from any distance sampling.

### 4.3.1 PDF derivation

To implement our sampling scheme into the Monte Carlo rendering framework, we need to compute the PDF of sampling a bridge. This is not a simple change of variables, since the dimensionalities do not match: We sample $n$ vertices, but we need a PDF for only the inner $n-1$ vertices $\bar{\mathbf{x}}_{1,n-1}$, since $\mathbf{x}_n$ is predetermined.

We approach this in two steps. First, we match the dimensionality by lifting $\bar{\mathbf{x}}_{1,n-1}$ to an auxiliary space. To do so, we pad $\bar{\mathbf{x}}_{1,n-1}$ with the auxiliary vertex $\mathbf{x}'_n$. This allows us to define a bijective map to the preliminary vertices $\bar{\mathbf{x}}'_{1,n}$, since their scale is determined by $\mathbf{x}'_n$. In the second step, we marginalize out $\mathbf{x}'_n$ and obtain the PDF on $\bar{\mathbf{x}}_{1,n-1}$.

**Mapping to the auxiliary space.** We define the bijective map from auxiliary space $\bar{X}\times X'_n$ to preliminary vertices $\bar{X}'$ by

$$m(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n) = \mathbf{x}_0 + \frac{r}{s}(R(\mathbf{x}_1 - \mathbf{x}_0), \dots, R(\mathbf{x}_n - \mathbf{x}_0)) = \bar{\mathbf{x}}'_{1,n}, \tag{4.1}$$

where $R$ is the shortest rotation from $\boldsymbol{\omega}_{0\to n}$ to $\boldsymbol{\omega}'_{0\to n}$ and $r$ and $s$ are the preliminary and the target connection distance, respectively:

$$r \coloneqq \left\|\mathbf{x}'_n - \mathbf{x}'_0\right\|_2, \quad s \coloneqq \|\mathbf{x}_n - \mathbf{x}_0\|_2. \tag{4.2}$$

Since $R$ is orthogonal and $\mathbf{x}'_n$ maps to itself, the Jacobian of $m$ is:

$$\det J_m(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n) = \begin{vmatrix} \operatorname{diag}\left(\frac{r}{s}, \dots, \frac{r}{s}\right) & * \\ 0 & \mathbf{I}_3 \end{vmatrix} = \left(\frac{r}{s}\right)^{3(n-1)}. \tag{4.3}$$

With the product PDFs of phase function sampling $p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n})$ and distance sampling $p_{\bar{D}'}\left(\bar{d}'_{1,n}\right)$, we can now derive the PDF in the auxiliary space $\bar{X}\times X'_n$, using $d'_i = \frac{r}{s}d_i$:

$$\begin{aligned} &p_{\bar{X}\times X'_n}(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n) = p_{\bar{X}'}(m(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n))\left|\det J_m(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n)\right| \\ &= p_{\bar{X}'}(\bar{\mathbf{x}}'_{1,n})\left(\frac{r}{s}\right)^{3(n-1)} \\ &= \left(\prod_{i=1}^{n}\frac{1}{(\frac{r}{s}d_i)^2}\right) p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n}) p_{\bar{D}'}\left(\frac{r}{s}\bar{d}_{1,n}\right)\left(\frac{r}{s}\right)^{3(n-1)} \\ &= \left(\prod_{i=1}^{n}\frac{1}{d_i^2}\right) p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n}) p_{\bar{D}'}\left(\frac{r}{s}\bar{d}_{1,n}\right)\left(\frac{r}{s}\right)^{n-3}. \end{aligned} \tag{4.4}$$

**Marginalization.** The remaining step is to marginalize out the auxiliary vertex $\mathbf{x}'_n$. We illustrate the configuration in Fig. 4.4. Since we chose $\boldsymbol{\omega}'_1 = \boldsymbol{\omega}_{0\to n}$ and $\boldsymbol{\omega}_1$ is fixed by $\mathbf{x}_1$, $R$ is already determined by $\boldsymbol{\omega}'_1 = R\boldsymbol{\omega}_1$. Note that this works only because of the symmetry of the construction: the shortest rotation between $\boldsymbol{\omega}'_1$ and $\boldsymbol{\omega}_1$ is the same as between $\boldsymbol{\omega}_{0\to n}$ and $\boldsymbol{\omega}'_{0\to n}$. With $R$ fixed, $\mathbf{x}'_n$ is on the line $\mathbf{x}'_n = \mathbf{x}_0 + rR\boldsymbol{\omega}_{0\to n}$ and we must only integrate over $r$. We multiply by $r^2$ for the change to the spherical

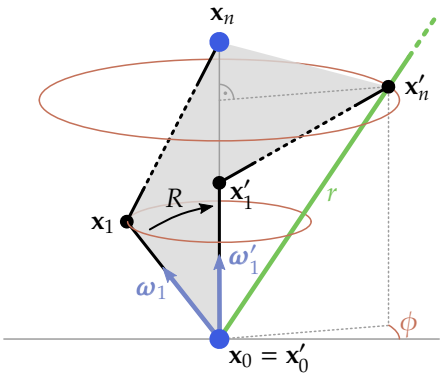



**Figure 4.4:** *For marginalization, we need to integrate over all $\mathbf{x}'_n$ which we could have sampled to construct the bridge $\bar{\mathbf{x}}_{0,n}$. Since the rotation $R$ is already determined by our fixed choice of $\boldsymbol{\omega}'_1$, $\mathbf{x}'_n$ is on a line (green) and the integration is just over radius $r$. The depicted $\mathbf{x}'_n$ is at $r = s$. Also note the rotational symmetry of the construction: for each value of $\phi$, the shortest rotation between $\boldsymbol{\omega}'_1$ and $\boldsymbol{\omega}_1$ is the same as between $\boldsymbol{\omega}_{0\to n}$ and $\boldsymbol{\omega}'_{0\to n}$.*

domain and obtain

$$\begin{aligned} p_{\bar{X}}(\bar{\mathbf{x}}_{1,n-1}) &= \int_{\Omega'} \int_0^\infty p_{\bar{X}\times X'_n}(\bar{\mathbf{x}}_{1,n-1}, \mathbf{x}'_n)\, r^2 \,\mathrm{d}r \,\mathrm{d}\delta_{\boldsymbol{\omega}_{0\to n}}(\boldsymbol{\omega}'_{0\to n}) \\ &= \left(\prod_{i=1}^{n} \frac{1}{d_i^2}\right) p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n}) \int_0^\infty p_{\bar{D}'}\left(\frac{r}{s}\,\bar{d}_{1,n}\right) \frac{r^{n-1}}{s^{n-3}}\,\mathrm{d}r. \end{aligned} \tag{4.5}$$

### 4.3.2 Distance sampling based on scaling

Next, we need to choose a specific distance sampling distribution. It should allow us to compute the value of the marginal integral and result in a favorable PDF. Motivated by free-flight sampling, we pick the exponential distribution with scale parameter $\lambda$

$$p_{\bar{D}'}\left(\bar{d}'_{1,n}\right) = \prod_{i=1}^{n} \lambda \exp(-\lambda d'_i) = \lambda^n \exp\left(-\lambda \left\|\bar{d}'_{1,n}\right\|_1\right), \tag{4.6}$$

and insert it into the marginal integral from Eq. (4.5):

$$\begin{aligned} &\int_0^\infty p_{\bar{D}'}\left(\frac{r}{s}\bar{d}_{1,n}\right) \frac{r^{n-1}}{s^{n-3}}\,\mathrm{d}r \\ &= \frac{\lambda^n}{s^{n-3}} \int_0^\infty \exp\left(-\frac{\lambda \left\|\bar{d}_{1,n}\right\|_1}{s} r\right) r^{n-1}\,\mathrm{d}r \\ &= \frac{\lambda^n}{s^{n-3}} (n-1)! \left(\frac{\lambda \left\|\bar{d}_{1,n}\right\|_1}{s}\right)^{-n} = \frac{s^3 (n-1)!}{\left\|\bar{d}_{1,n}\right\|_1^n}. \end{aligned} \tag{4.7}$$

This leads to Algorithm 1 and the PDF in product vertex area measure:

$$\boxed{p_{\bar{X}}\left(\bar{\mathbf{x}}_{1,n-1}\right) = \left(\prod_{i=1}^{n} \frac{1}{d_i^2}\right) p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n}) \frac{s^3 (n-1)!}{\left\|\bar{d}_{1,n}\right\|_1^n}.} \tag{4.8}$$

**Algorithm 1** Sample bridge from $\mathbf{x}_0$ to $\mathbf{x}_n$

$n \leftarrow$ sample vertex count with probability $P_n$ ▷ *see Section 4.4*
$\mathbf{x}'_0 \leftarrow \mathbf{x}_0$
$p \leftarrow P_n$
$\boldsymbol{\omega}'_1 \leftarrow \boldsymbol{\omega}_{0\to n}$
**for** $i \leftarrow 1, \dots, n$ **do**
  **if** i > 1 **then**
    $\boldsymbol{\omega}'_i \leftarrow$ sample phase function given $\boldsymbol{\omega}'_{i-1}$
    $p \leftarrow p\, p_\Omega(\boldsymbol{\omega}'_i)$ ▷ *phase function PDF*
  $\bar{d}'_i \leftarrow -\ln(1 - \text{rand}())$ ▷ *rand() ∈ [0, 1)*
  $\mathbf{x}'_i \leftarrow \mathbf{x}'_{i-1} + d'_i \boldsymbol{\omega}'_i$
compute rotation $R$ such that $\mathbf{x}'_n - \mathbf{x}_0 = \frac{\|\mathbf{x}'_n - \mathbf{x}_0\|_2}{\|\mathbf{x}_n - \mathbf{x}_0\|_2} R(\mathbf{x}_n - \mathbf{x}_0)$
**for** $i \leftarrow 1, \dots, n$ **do**
  $\mathbf{x}_i \leftarrow \mathbf{x}_0 + \frac{\|\mathbf{x}_n - \mathbf{x}_0\|_2}{\|\mathbf{x}'_n - \mathbf{x}_0\|_2} R^T(\mathbf{x}'_i - \mathbf{x}_0)$
  $\boldsymbol{\omega}_i \leftarrow R^T \boldsymbol{\omega}'_i$
  $d_i \leftarrow \frac{\|\mathbf{x}_n - \mathbf{x}_0\|_2}{\|\mathbf{x}'_n - \mathbf{x}_0\|_2} d'_i$
logp $\leftarrow$ lgamma(n) $- n \ln \sum_{i=1}^n d_i$ ▷ *log space to handle large n*
$p \leftarrow p\, \|\mathbf{x}_n - \mathbf{x}_0\|_2^3 \exp(\text{logp})$ ▷ *product solid angle PDF*

**Discussion.** The scale parameter $\lambda$ does not appear in the result, which means that sampling of preliminary distances is scale-invariant in this case. This indicates that the result is not related to free flights (see Appendix 4.A for an alternative view) and we cannot use it to sample transmittance. Still, as we will show, the resulting PDF is suitable for our purposes. We canceled part of the $s^{-(n-3)}$ term, with only a constant $s^3$ remaining. The total traveled distance $\|\bar{d}_{1,n}\|_1$ has low variance with strong forward scattering, and the factorial varies only with bridge length.

### 4.3.3 Constraint-based distance sampling for $n = 2$ and $n = 3$

With scaling-based sampling, we still incur a non-constant Jacobian, which is an additional source of variance. The Jacobian occurs, since we take infeasible paths, i.e. with unfit $\|\mathbf{x}'_n - \mathbf{x}'_0\|_2$, and project them to feasible paths that achieve the fixed connection distance $\|\mathbf{x}_n - \mathbf{x}_0\|_2$. In this section, our goal is to

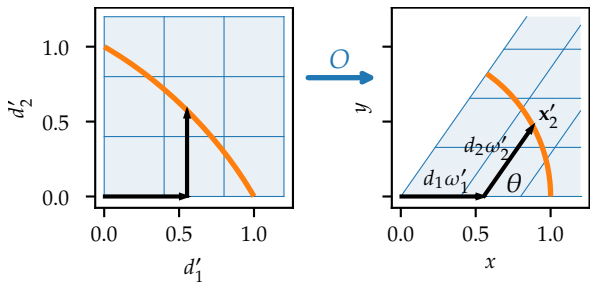



**Figure 4.5:** *Constraint-based sampling for $n = 2$ with $\omega'_1, \omega'_2$ in the $xy$-plane and $\omega'_1 \cdot \omega'_2 = \cos\theta$. $O$ maps a pair of distances $d'_1, d'_2$ to $x'_2$. The feasible region (orange), i.e. where $d'_1, d'_2 > 0$ and in this example $\|O\bar{d}'\|_2 = 1$, is an arc of length $\theta$. We can sample valid distances uniformly by first sampling a point on this arc uniformly and then solving for distances using $O$.*

uniformly sample the space of feasible paths to get rid of the Jacobian term. We achieve this for the case of one or two inserted vertices, i.e. $n = 2$ or $n = 3$, and show how generalization to higher dimensions remains difficult.

**Space of feasible paths.** To describe the space of feasible paths, we define the linear map $O$ from preliminary vertex distances $\bar{d}'$ to the vector $\mathbf{x}'_n - \mathbf{x}'_0$, i.e. $\mathbf{x}'_n - \mathbf{x}'_0 = O\bar{d}'$:

$$O = \begin{pmatrix} | & | & & | \\ \boldsymbol{\omega}'_1 & \boldsymbol{\omega}'_2 & \dots & \boldsymbol{\omega}'_n \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{3\times n}. \tag{4.9}$$

This uses the edge directions $\boldsymbol{\omega}'_i$ that are already determined by phase function sampling. $\bar{d}'$ produces a feasible path if it satisfies the constraints on

1. connection distance $\left\|O\bar{d}'\right\|_2 = \|\mathbf{x}_n - \mathbf{x}_0\|_2 = s$,
2. distance positivity $\forall i \in \{1, \dots, n\}\colon d'_i > 0$.

**Algorithm idea.** The first constraint tells us that $O$ maps all feasible distances to a region on the sphere (or circle if $n = 2$) with radius $s = \|\mathbf{x}_n - \mathbf{x}_0\|_2$, which we call the feasible region. The distance positivity constraint restricts the feasible region to an intersection of $n$ halfspaces, which results in an arc for $n = 2$ (see Fig. 4.5) and a spherical triangle for $n = 3$. If $O$ is invertible, we can pick a random point $\mathbf{y}$ in the feasible region defined by $O^{-1}\mathbf{y} > 0$ and solve for a distance vector $\bar{d}' = O^{-1}\mathbf{y}$. Unfortunately, this is in general not the case, as $\operatorname{rank}(O) \leq 3$, but we can generalize our idea using the singular value decomposition (SVD).

**Sampling algorithm.** We describe our algorithm for the general case, even though we will later show that it achieves the desired PDF only for $n = 2$ and $n = 3$. First, we compute an SVD of $O = U\Sigma V^T$, i.e. $U \in \mathbb{R}^{3\times 3}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $k = \operatorname{rank}(O)$ and

$$\Sigma = \begin{pmatrix} \tilde{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{3\times n}, \quad \tilde{\Sigma} = \operatorname{diag}(\sigma_1, \dots, \sigma_k). \tag{4.10}$$

We refer to the column vectors of $U$ and $V$ as $u_i$ and $v_i$. The image of $O$ is $\operatorname{im}(O) = \langle u_1, \dots, u_k \rangle$ and its null space is $\ker(O) = \langle v_{k+1}, \dots, v_n \rangle$.

Our idea of choosing a random point $\mathbf{y}$ in the feasible region of the image $\operatorname{im}(O)$ followed by inversion is still applicable, only that the inversion is generally ambiguous. We need to additionally sample a point in the feasible region of the null space $\ker(O)$ if $k < n$. To sample $\bar{d}$, we follow these steps:

1. Sample $\mathbf{y}' \in \mathbb{R}^k$ uniformly in the feasible region of the image $\operatorname{im}(O)$.
2. Set $\mathbf{y}_{1,\dots,k} \coloneqq \tilde{\Sigma}^{-1}\mathbf{y}'$.
3. Sample $\mathbf{y}_{k+1,\dots,n}$ uniformly in the feasible region of the null space $\ker(O)$.
4. Set $\bar{d} = V\mathbf{y}$.

We can define both feasible regions again using inequalities on the sampled distances, analogous to the invertible case.

**PDF derivation.** We can use the result from Eq. (4.5) and insert the value of the marginal integral. Since in this case $r = s$, the integral vanishes completely, and we simplify using $\frac{r}{s} = 1$:

$$\begin{aligned}
&\int_0^\infty p_{\bar{D}'}\left(\frac{r}{s}\bar{d}_{1,n}\right)\frac{r^{n-1}}{s^{n-3}}\,\mathrm{d}\delta_s(r) = p_{\bar{D}'}\left(\bar{d}_{1,n}\right)s^2 \\
&= p_{Y_{1,\dots,k}}\left(\mathbf{y}_{1,\dots,k}\right)p_{Y_{k+1,\dots,n}}\left(\mathbf{y}_{k+1,\dots,n} \mid \mathbf{y}_{1,\dots,k}\right)s^2 \\
&= p_{Y'_{1,\dots,k}}\left(\mathbf{y}_{1,\dots,k}\right)\left(\prod_{i=1}^{k}\sigma_i\right)p_{Y_{k+1,\dots,n}}\left(\mathbf{y}_{k+1,\dots,n} \mid \mathbf{y}_{1,\dots,k}\right)s^2 \\
&= \frac{1}{V_{\text{img}}}\left(\prod_{i=1}^{k}\sigma_i\right)\frac{1}{V_{\text{null}}(\mathbf{y}_{1,\dots,k})}s^2,
\end{aligned} \tag{4.11}$$

where we define $V_{\text{img}}$ and $V_{\text{null}}$ as the volumes of the feasible regions in the image and null space, respectively. If $n = 2$, this PDF is equivalent to OMNEE [HWD22] (see Appendix 4.B).

**Discussion.** Our sampling achieves the goal of keeping the part of the PDF in Eq. (4.11) constant if $k = n$. Otherwise, sampling the null space introduces a non-constant term, since the volume of its feasible region depends on $\mathbf{y}_{1,\dots,k}$. Therefore, we would have to sample $\mathbf{y}_{1,\dots,k}$ non-uniformly in order to cancel this term. In addition, just computing the volume of a feasible region in null space quickly turns intractable: the general case of computing the volume of a halfspace intersection in $n$ dimensions is NP-hard [DF88]. In summary, this approach does not seem promising for scaling to many vertices.

## 4.4 Sampling the number of vertices

One key advantage of bridge sampling is that it provides the freedom to choose the vertex count for a connecting subpath. So far, we only addressed sampling bridges of predetermined vertex counts, leaving aside the question of choosing a suitable vertex count.

The reference for vertex count is the analog scattering process in the medium: We would like to match the distribution of the number of bridge vertices to the collision count of paths in the medium weighted by their contribution. Several medium properties influence this distribution. High density in relation to connection distance leads to more collision events on average. Losses of contribution due to these collisions are reduced by a high scattering albedo (reduced absorption) and a strong forward scattering phase function (reduced out-scattering).

The sampling decision for bridge vertex counts needs to consider these effects subject to medium properties. This is not a simple task, even when assuming a homogeneous medium. For example, in transmittance estimation, collision along a connection in the medium is described by a Poisson process [GMH*19]. This tells us the distribution of collision counts for the given track length, i.e. the connection distance. In our case, the track length is not known in advance, since it is itself a result of our bridge sampling.

Therefore, we resort to precomputing the expected contribution in a homogeneous medium. For each vertex count and phase function parameter, we store the precomputed contribution as a 1D function of the product $s\mu_t$. During sampling, we recover a distribution for sampling by evaluating this function for each possible vertex count and normalizing to the sum (see Algorithm 2). In the following, we explain our precomputation. Also see Appendix 4.C for an alternative approach known from the neutron transport literature, where vertex count is chosen implicitly as the result of a random walk.

### 4.4.1 Defining the bridge estimator

We define the $n$-bridge estimator that computes the transport between two vertices $\mathbf{a} = \mathbf{x}_0$ and $\mathbf{b} = \mathbf{x}_n$ with $n-1$ scattering events in between as:

$$B_n = \frac{f_{\text{bridge}}(\bar{\mathbf{x}}_{1,n-1})}{p_{\bar{X}}(\bar{\mathbf{x}}_{1,n-1})}, \quad B_1 = \frac{T(\mathbf{a},\mathbf{b})}{s^2} \tag{4.12}$$

where $f_{\text{bridge}}$ is the measurement contribution for the bridge subpath without phase functions and cosines at $\mathbf{a}$ and $\mathbf{b}$:

$$f_{\text{bridge}}(\bar{\mathbf{x}}_{1,n-1}) = \left(\prod_{i=1}^{n} \frac{T(\mathbf{x}_{i-1},\mathbf{x}_i)}{d_i^2}\right)\left(\prod_{i=1}^{n-1} \mu_s f_s(\ldots)\right). \tag{4.13}$$

$B_1$ introduces no additional vertices and represents the deterministic connection done by conventional next-event estimation. For the complete transport, we sum over all estimators: $\mathcal{B}^\infty = \sum_{n=1}^{\infty} B_n$. As the contribution of many $B_n$ will be quite low, depending on the configuration, our goal is to stochastically evaluate only a single term with probability $P_n$:

$$\mathcal{B}^1 = \frac{B_n}{P_n}. \tag{4.14}$$

This is a good estimator of the sum, as each $B_n$ has low variance individually. To handle the residual variance of $B_n$, we target the square root of its second moment [VKK18, Appendix B]:

$$P_n \sim \sqrt{\mathbb{E}[B_n^2]} \tag{4.15}$$

by precomputing it for homogeneous media.

### 4.4.2 Precomputing $\sqrt{\mathbb{E}[B_n^2]}$ for homogeneous media

With the assumption of a homogeneous medium, we can precompute an approximation to $\sqrt{\mathbb{E}[B_n^2]}$ for use as $P_n$, which is the probability for constructing $n-1$ inner bridge vertices. We assume an infinite medium with constant $\mu_s$, $\mu_t$, and phase function, as well as full visibility and perfect importance sampling of the phase function. Note that our assumptions do not have to strictly hold in practice, since we use them for importance sampling only. Together with the PDF from Eq. (4.8), $B_n$ reduces to:

$$B_n = \frac{\mu_s^{n-1} \left\|\bar{d}_{1,n}\right\|_1^n \exp\left(-\mu_t \left\|\bar{d}_{1,n}\right\|_1\right)}{s^3 (n-1)!}. \tag{4.16}$$

The only remaining random variable in the estimator is the total track length of the bridge $\left\|\bar{d}_{1,n}\right\|_1 =: D_\Sigma$. By linearity, we can thus reduce $\sqrt{\mathbb{E}[B_n^2]}$ to the expectation of a function $b_n$ of $\mu_t D_\Sigma$:

$$\begin{aligned}\sqrt{\mathbb{E}[B_n^2]} &= \sqrt{\mathbb{E}\left[\left(\frac{\mu_s^{n-1} D_\Sigma^n \exp\left(-\mu_t D_\Sigma\right)}{s^3(n-1)!}\right)^2\right]} \\ &= \frac{\mu_s^{n-1}\sqrt{\mathbb{E}\left[b_n\left(\mu_t D_\Sigma\right)^2\right]}}{\mu_t^n s^3 (n-1)!},\end{aligned} \tag{4.17}$$

with $b_n(x) \coloneqq x^n e^{-x}$. The distribution of $D_\Sigma$ depends on the phase function, number of vertices $n$, and the connection distance $s$. For each combination of $2 \le n \le n_{\max}$ and phase function parameters, we precompute and store an approximation to the 1D function

$$\tilde{b}_n(s\mu_t) = \frac{\sqrt{\mathbb{E}\left[b_n(s\mu_t D'_\Sigma)^2\right]}}{(n-1)!}, \tag{4.18}$$

where $D'_\Sigma \coloneqq s^{-1} D_\Sigma$ is the normalized track length. In our computed approximations, we always use the Henyey-Greenstein phase function. The resulting functions tend to look very smooth (see Fig. 4.6) and we approximate each function using a single cubic spline. We use Monte Carlo integration with our bridge sampling to evaluate Eq. (4.18) during precomputation.

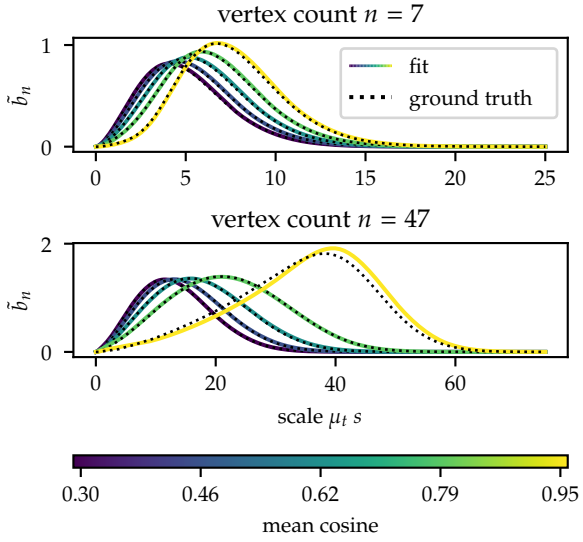



**Figure 4.6:** *$\tilde{b}_n$ and our stored approximation for two different vertex counts and different mean cosines. With increasing mean cosine, its peak shifts toward $s\mu_t = n$. The function is smooth and mostly approximated well by a spline.*

### 4.4.3 Importance sampling $n$

To recover a probability distribution for sampling $n$, we follow Algorithm 2. We evaluate Eq. (4.17) using the spline approximation and medium parameters for each $n$ up to $n_{\max}$ and normalize. We show vertex count distributions computed by our algorithm in Fig. 4.7. We can choose between bridge sampling and next-event estimation by including $n = 1$ with $\sqrt{\mathbb{E}[B_1^2]} = B_1$ (see Eq. (4.12)).

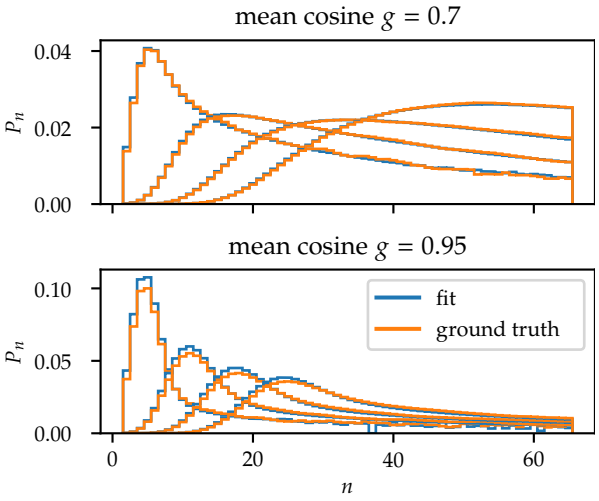



**Figure 4.7:** *Our approximated distribution for importance sampling n and its reference distribution. We show the distribution evaluated with albedo 0.99 for different scale parameters $s\mu_t = 4, 10, 16, 22$, where the peak shifts to the right with increasing scale. The less peaked $g = 0.7$ leads to longer track length on average and therefore selects higher vertex counts. A lower albedo would reduce the contribution of these paths and thus the tail of the distribution. Note that we omit the next-event estimation $n = 1$ case in this plot.*

**Algorithm 2** Sample path length

$\tilde{P}_1 \leftarrow \frac{T(\mathbf{a},\mathbf{b})}{s^2}$ ▷ *see Eq.* (4.12)
norm $\leftarrow \tilde{P}_1$
**for** $n = 2, \ldots, n_{\max}$ **do**
  spline $\leftarrow$ lookup$(n, g)$
  $\tilde{P}_n \leftarrow \frac{\mu_s^{n-1}}{\mu_t^n s^3}$spline$(s\mu_t)$ ▷ *see Eq.* (4.17)
  norm $\leftarrow$ norm $+\tilde{P}_n$
select $n$ with probability $P_n = \frac{\tilde{P}_n}{\text{norm}}$

## 4.5 Results

### 4.5.1 Parametric unit test case

We evaluate our method primarily on a parametric unit test case, for which we provide source code [SHD24b]. Our test is a point light source in an infinite homogeneous medium. We use the Henyey-Greenstein phase function with mean cosine $g$. The scattering albedo is set to 0.99 in our test cases to emphasize multiple scattering. We parameterize the medium extinction by specifying $s\mu_t$, which is the

expected number of collisions along the distance $s$ between camera and light source. By default, we use our scaling-based distance sampling for bridges. In our results, we use $k$ to refer to the path length as number of vertices, which includes camera and light source. We abbreviate the number of samples per pixel as spp.

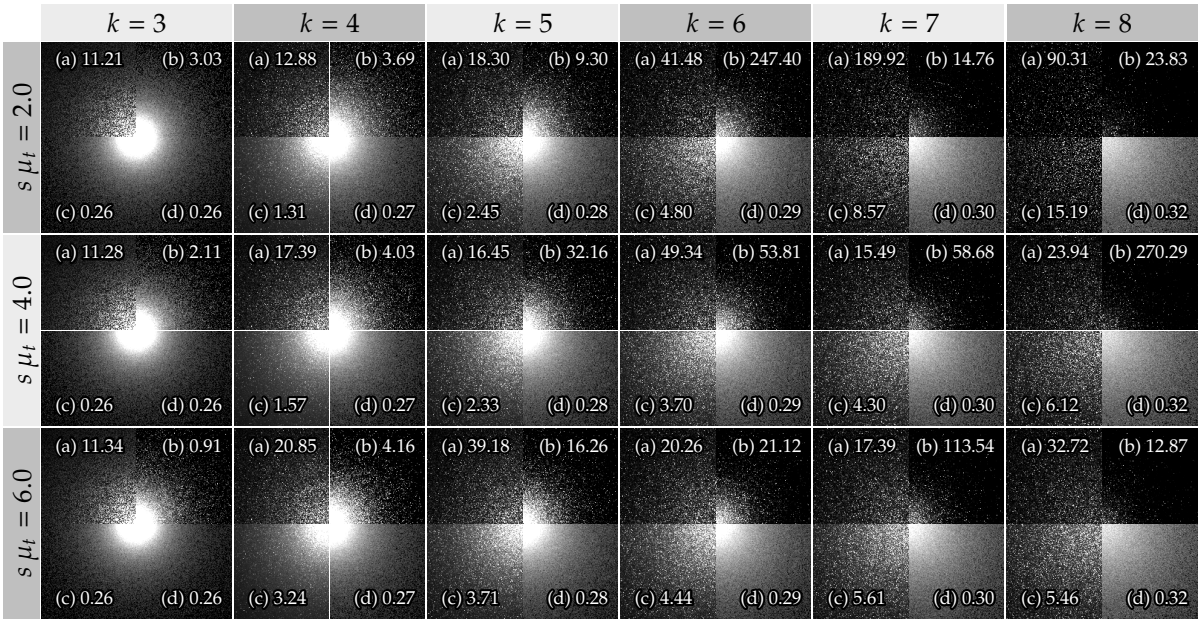
**Normalization and errors.** For comparison to reference images, we use a relative root mean squared error (rRMSE), which is the RMSE normalized by mean image brightness of the reference. We also expose all renderings such that the reference achieves a fixed mean brightness. This is to make noise patterns better comparable between different sets of parameters. The numbers next to inset labels in our result figures are always rRMSE errors.

**Comparison to related work.** We compare our bridge sampling to other methods for extended next-event estimation in volumes: equiangular sampling, MVNEE and OMNEE. In Fig. 4.8, we show the behavior of these methods in a strongly forward-scattering medium ($g = 0.95$) with different combinations of medium extinctions and path lengths. In Fig. 4.9, we show a similar comparison with fixed extinction, but different combinations of mean cosines and path length. These comparisons include only the forward-scattering part of the phase function, as OMNEE and MVNEE do not handle backscattering. Our bridge sampling performs well consistently in all these configurations. While OMNEE and MVNEE also show good results for single scattering, they cannot sample longer paths well. For OMNEE and equiangular, this is due to the reliance on path tracing for $k > 3$. MVNEE explicitly constructs longer connecting subpaths, similar to our method. However, since it perturbs vertices independently, its sampling does not reflect important vertex correlations caused by phase functions and geometry terms. Our bridge sampling shares the slight degradation of performance toward lower mean cosine. Since it targets the case of pronounced forward scattering, this is expected and generally not concerning. We list render times in Table 4.2. The runtime performance of our method is comparable to previous methods, with an overhead over path tracing with equiangular sampling of up to 2×. Note that this test case might not reflect performance in a full renderer, due to the absence of visibility tests and our implementation in Python, which is JIT-compiled and executed on the GPU through JAX.

**Table 4.2:** *Render times for computing the first row of Fig. 4.8.*

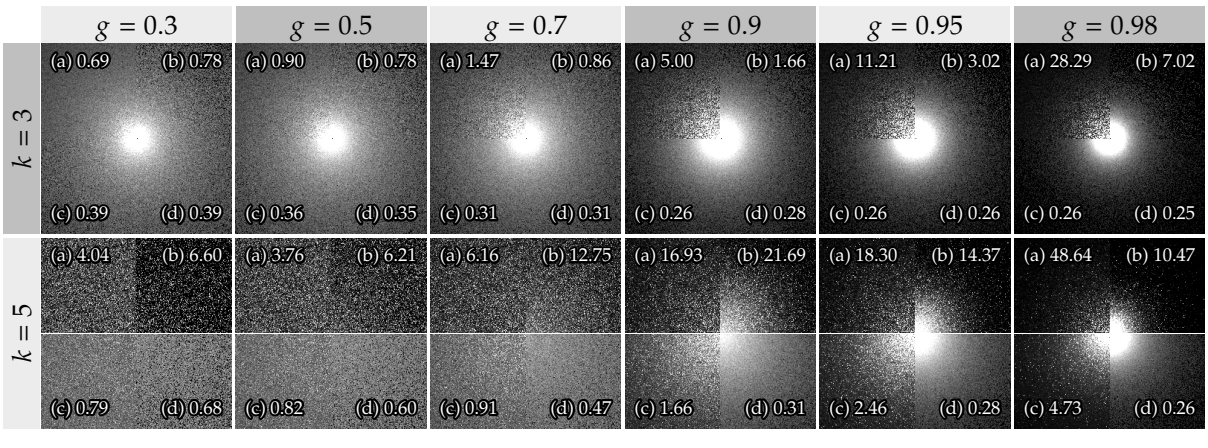
| **time (ms)** | $k = 3$ | $k = 4$ | $k = 5$ | $k = 6$ | $k = 7$ |
|---|---|---|---|---|---|
| Equiangular | 5.89 | 6.55 | 7.43 | 8.59 | 9.80 |
| MVNEE | 8.89 | 11.33 | 13.65 | 15.59 | 16.49 |
| OMNEE | 8.90 | 7.13 | 7.22 | 8.74 | 9.07 |
| Bridges | 12.22 | 11.87 | 12.79 | 16.38 | 18.26 |

**Scaling-based and constraint-based sampling.** In Fig. 4.10, we compare our scaling-based sampling (see Section 4.3.2) with our theoretically superior constraint-based sampling (see Section 4.3.3). We demonstrate lower standard deviation of the constraint-based estimator when not considering transmittance. However, we do not observe this difference in practice, even when rendering a thin medium. This can be attributed to the difference in the distribution of sampled track lengths as shown in Fig. 4.11.



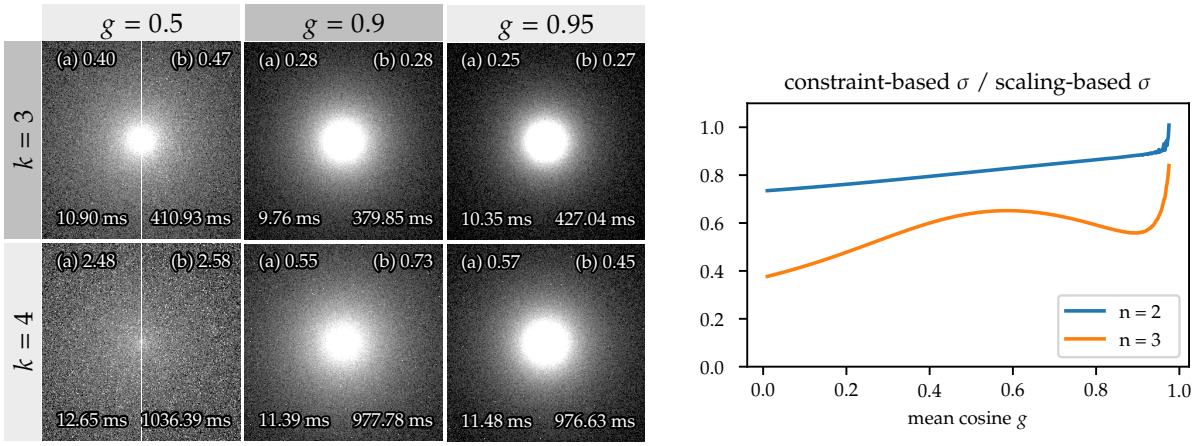

(a) Equiangular (b) MVNEE (c) OMNEE (d) Bridges

**Figure 4.8:** *Comparison of our sampling with different extinctions $\mu_t$ and path lengths $k$, $g = 0.95$, and 16 spp. Equiangular sampling struggles with the peaked phase function, while all other methods handle the single scattering case $k = 3$ well. For $k > 3$, equiangular and OMNEE have to rely on path tracing a prefix, worsening their results with longer paths. MVNEE also struggles to construct longer paths, since its independent perturbations neglect vertex correlation. Our bridge sampling performs well consistently, with the error deteriorating only slightly toward longer paths.*




(a) Equiangular (b) MVNEE (c) OMNEE (d) Bridges

**Figure 4.9:** *Comparison of our sampling with different mean cosines $g$ and path lengths $k$, $s\mu_t = 2$ and 16 spp. Equiangular sampling handles moderately forward scattering well, but cannot deal with highly peaked phase functions. Similar to OMNEE, the variance of our bridge sampling increases with less pronounced forward scattering, due to an increased variance in the sampled track lengths. This also lets variance increase faster with path length $k$. While this is generally not a problem, as our method is targeted at strong forward scattering, we still achieve relatively low errors consistently.*




(a) Scaling-based (b) Constraint-based

**Figure 4.10:** *Comparison of our scaling-based bridge sampling against our constraint-based sampling with $n = 2$ ($k = 3$) and $n = 3$ ($k = 4$). Left: Renderings with 16 spp in a thin medium with $s\mu_t = 0.1$. Although constraint-based sampling sometimes achieves slightly lower error, differences are negligible. Right: We compare the ratio of standard deviations $\sigma$ of both estimators without transmittance as a function of mean cosine $g$. In this theoretical comparison, constraint-based sampling achieves distinctly lower standard deviation especially for low $g$, but the ratio approaches 1 toward $g = 1$.*

Depending on the angular configuration, constraint-based sampling can lead to much longer track lengths on average, increasing transmittance variance. Therefore, it tends to lose its theoretical benefit if transmittance is significant. An intuitive explanation for this behavior is that in comparison to constraint-based sampling, our scaling-based approach is closer to sampling the transmittance term.

**Forward-only and full sphere phase function.** We examine the variance increase that comes with enabling backwards scattering in our method in Fig. 4.12. This usually has only a very moderate effect, especially for higher values of $g$. Even for these, bridge sampling can sometimes construct paths with large track length when backward scattering is enabled. These would produce outlier samples because

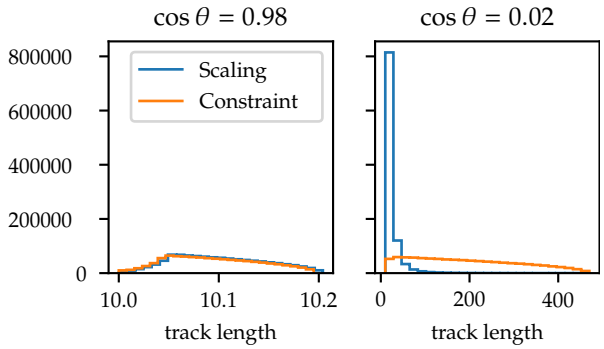
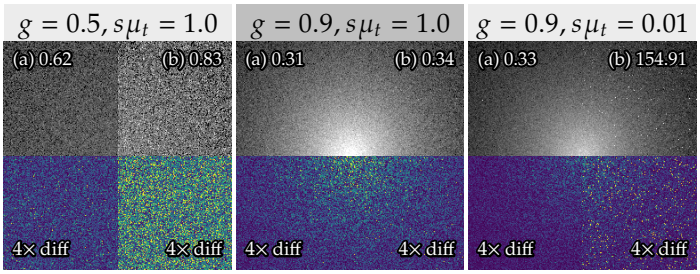



**Figure 4.11:** *Histograms of sampled total track lengths for two different angular configurations of bridges with 3 edges in a plane and $s = \|\mathbf{x}_n - \mathbf{x}_0\|_2 = 10$. With near-parallel directions (left, $\cos\theta = 0.98$), the track length is close to $s$ for both sampling methods. Constraint-based sampling leads to a much wider distribution of track lengths with near-orthogonal directions (right, $\cos\theta = 0.02$).*




(a) Forward only (b) Forward + backward

**Figure 4.12:** *Enabling backward scattering leads to a slight increase in variance with $g = 0.5$. With stronger forward scattering ($g = 0.9$), the backwards hemisphere has only negligible extra contribution. In a very thin medium with $s\mu_t = 0.01$, long paths caused by backwards scattering produce outlier samples, best visible in the difference image. For this comparison, we use a path length of $k = 5$, 16 spp, and the same exposure for (a) and (b).*

of their low PDF, but this effect is usually counterbalanced by their low transmittance. In extremely thin media this is not the case and bridge sampling produces outlier samples.

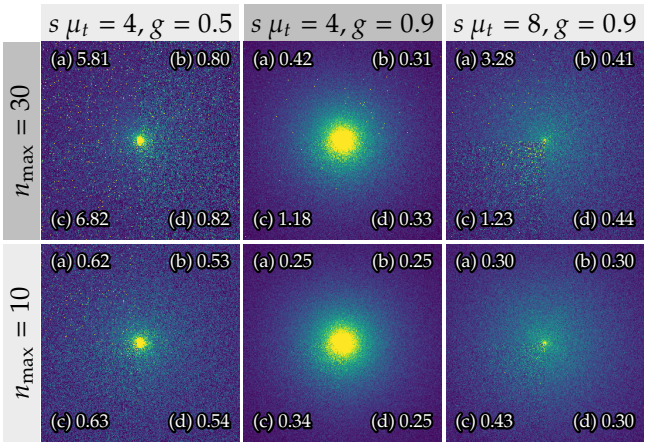
**Vertex count sampling.** We evaluate our method for sampling the vertex count of a bridge in Fig. 4.13. Besides our presented second moment-based method, we compare against using the first moment in its place, and the Poisson distribution, which is the method used by MVNEE. Since Poisson tends to have a light tail, we further compare against a version of Poisson with a scaled parameter. Sampling the vertex count well seems to matter only starting at longer paths, i.e. above 10 vertices. In these cases, our sampling outperforms Poisson, which starts to show outliers. Our second-moment-based sampling shows similar results to our first-moment-based sampling, although it can be more robust in thinner media.

**Scaling to long paths.** We examine the scalability of our approach to long paths up to $k = 100$ vertices in Fig. 4.14. While there is noticeable variance increase with more vertices, our sampling is still usable, while path tracing with equiangular connections hardly finds any contributing paths. Nevertheless, even with $g = 0.98$ the distribution of radiance looks very diffuse after so many scattering events, casting doubt on the importance of sampling the exact visibility in these cases.

### 4.5.2 Renderer integration and heterogeneous media

We also implemented our bridge sampling in a spectral renderer with support for heterogeneous media. To construct a bridge, we read $\mu_t$ at the endpoint of the current path and use this to run Algorithm 2. We continue to construct a tentative path following Algorithm 1, assuming a homogeneous medium with unit extinction. That is, in this step we do not evaluate the heterogeneous medium at path edges or vertices. After path construction is finished, we evaluate the transmittances along the path to compute the measurement contribution. This includes testing each edge for visibility. Unlike in Section 4.5.1, we do not directly connect camera and light sources via bridges in this implementation. Therefore, the single-scattering contribution does not benefit from our method in the following comparisons.

In Fig. 4.1, we show a medium with density varying in space and time (for motion blur). We

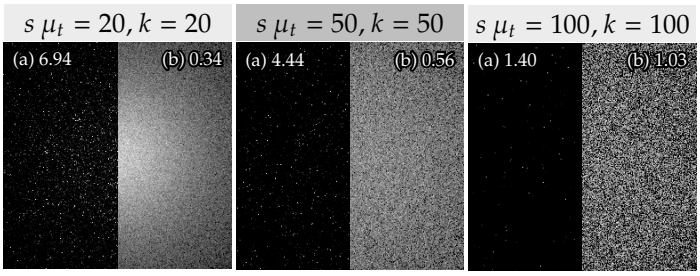



(a) Poisson $\lambda = s\mu_t$ (b) Ours first moment
(c) Poisson $\lambda = 1.5s\mu_t$ (d) Ours second moment

**Figure 4.13:** *Comparison of different strategies for selecting vertex counts for bridge sampling with 32 spp. We show this comparison in false-color to highlight outlier samples. Significant differences emerge only for many vertices ($n_{max} > 10$). Poisson sampling undersamples these bridges with many vertices, even with a scaled parameter, causing outliers. Using our first moment-based sampling largely mitigates this problem, although only our second moment-based sampling is robust in the case with $s\mu_t = 4$, $g = 0.9$ (top row, center).*

approximate Mie scattering with a 50% mixture of two Henyey-Greenstein lobes with strong forward scattering mean cosine of $g = 0.98$ and a backward lobe of $g = -0.4$ to show performance of the algorithm also in the presence of backward scattering. Our sampling technique provides a clear benefit over standard volumetric path tracing with next-event estimation. Note that this setup is intentionally challenging for our algorithm since it contains spot emitters with non-diffuse angular characteristics, heterogeneous volume properties, backward scattering, and occlusion, violating all assumptions of our technique.

Figures 4.15 and 4.16 show a scene with an occluder casting light shafts into a medium with varying parameters. We compare our method (Bridges) against path tracing with next-event estimation (PTNEE) and bidirectional path tracing (BDPT). BDPT handles moderate forward scattering ($g = 0.5$) well, but struggles with strongly peaked phase functions ($g = 0.95$). The sampling quality of our method




(a) Equiangular (b) Bridges

**Figure 4.14:** *Our bridge sampling can scale to very long paths, as shown in this comparison for paths with up to $k = 100$ vertices, $g = 0.98$ and 16 spp. Variance eventually increases with $k$, but remains at a moderate level. Note that the error for equiangular is unreliable here, as it produces only few samples with significant contribution.*

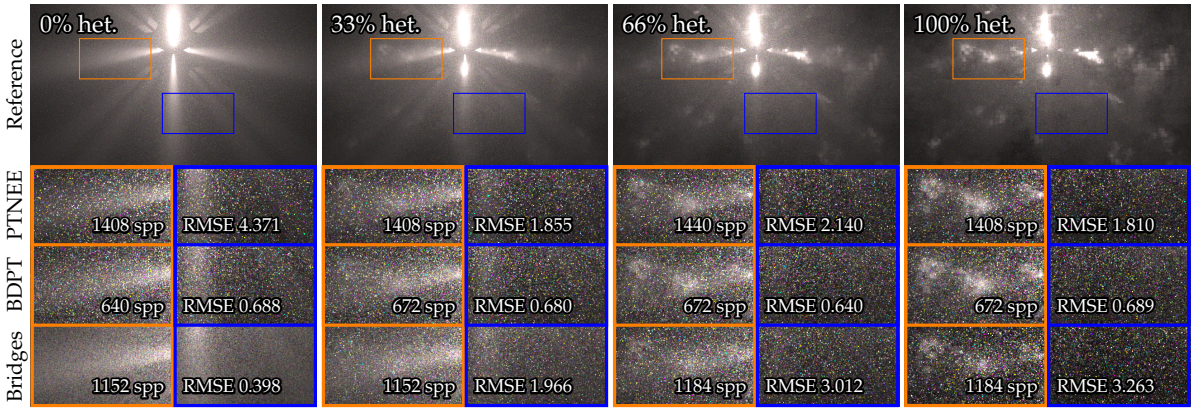



**Figure 4.15:** *A partially shadowed light source in a dense, forward-scattering ($g = 0.95$) medium of varying heterogeneity (het.), rendered in equal time (180s). Our bridge sampling handles the multiple-scattering contribution well in homogeneous media, but deteriorates with increasing heterogeneity.*

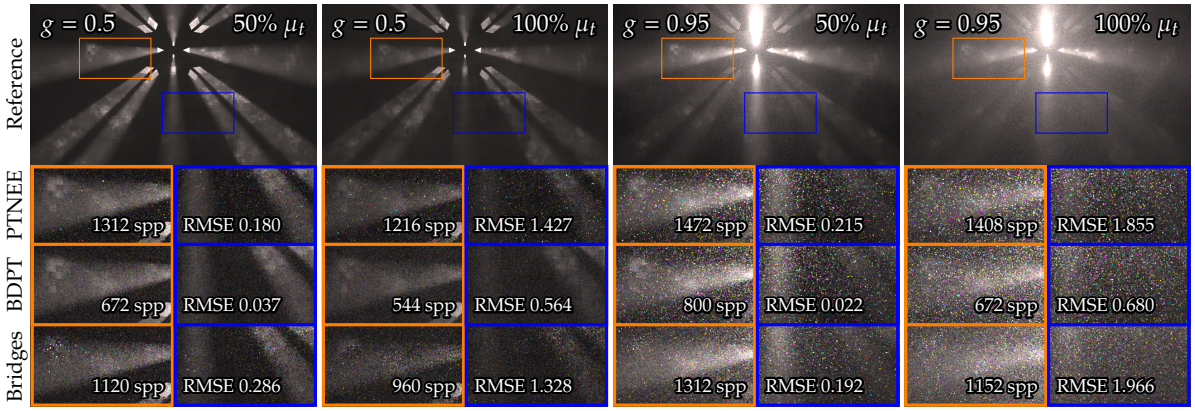



**Figure 4.16:** *Equal-time comparison (180s) for different parameters of the scene from Fig. 4.15 with 33% heterogeneity. While BDPT does not handle strongly peaked phase functions ($g = 0.95$) well, it outperforms our sampling with less pronounced forward scattering ($g = 0.5$). Likewise, in thinner media ($50\%\mu_t$), multiple scattering is not as distinct and our method shows less improvement.*

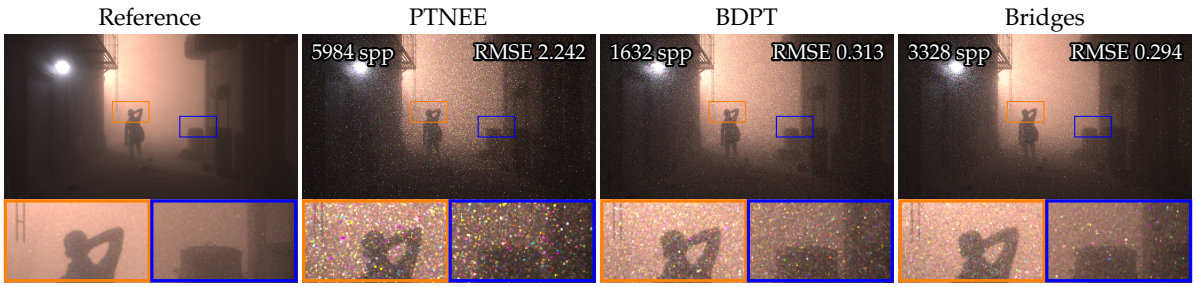



**Figure 4.17:** *Equal-time comparison (600s) with more complex geometry. The alley is mainly lit by a street lamp behind the corner and filled with dense, homogeneous fog (HG mixture of $g = 0.98$ and $g = -0.4$), leading to long paths with significant contribution. Our method is slowed down by its increased number of visibility tests, but still outperforms PTNEE and BDPT due to improved sampling quality.*

deteriorates with increasing medium heterogeneity. One issue here might be our simplistic path length sampling, which is based on the end vertex density. This leads to sampling longer paths than necessary in dense regions of the medium.

**Overhead.** Our method comes with some runtime overhead as compared to simple next-event estimation. First, multiple edges will have to be tested for visibility, incurring an extra cost in the ray tracing when initially traversing the BVH from the root. Second, computing the transmittance in a heterogeneous medium has a cost for each voxel that is accessed. Since next-event estimation steps through voxels along the direct connection, this is going to be cheaper than our longer track lengths.

To match the runtime of simple volumetric path tracing in Fig. 4.1, we use a Russian roulette decision whether to construct a bridge at an end vertex or not, with a survival probability of 30%. That is, for this maximum path length of 10, sampling bridges is about 3× as expensive as standard next-event estimation. The probability of 30% turned out to result in about equal rendering time for a fixed sample count in this scene, in turn increasing variance. Practical applications should rather base this decision on a more global argument whether or not a connection is expected to yield a good contribution [VK16].

In Fig. 4.17, we show our method on a scene with more interesting geometry and visibility. We render with a maximum path length of 14 vertices. Even though the medium is homogeneous in this scene, the overhead of our method is higher than in Figs. 4.15 and 4.16 due to higher cost of visibility tests. Nevertheless, our method achieves lower error in an equal-time comparison against PTNEE and BDPT.

## 4.6 Conclusion and future work

We presented *bridge sampling*, a technique to connect light transport path endpoints to light sources via multiple scattering and an arbitrary number of intermediate vertices. This technique can sample all the reciprocal squared distance and internal phase function terms, for arbitrary phase functions (including both forward and backward scattering). We showed how this works well for a variety of parameters in homogeneous volumes, up to 100 extra vertices, and compared against current volume sampling techniques specializing in peaked forward scattering. The technique is simple to implement and, while based on consideration of homogeneous volumes, has a clear benefit when applied to heterogeneous volumes.

The resulting PDF does not consider the phase function or emission distribution function at either endpoint of the bridge. The phase function can easily be included by the same strategy as standard next-event estimation: in addition to sampling a bridge, also extend the path by sampling the phase function. Including the emitter term may be an interesting future problem to consider, especially for strongly directional spot lights. One approach could be to generalize our method by using it to form connections in bidirectional path tracing. This could also enable further applications of our method where the light source is outside of the medium, e.g. to subsurface scattering or rendering of clouds.

Since the marginalization over $\mathbf{x}_n$ worked out analytically, in the future it might be possible to use this approach to derive a closed-form formulation of the transport in homogeneous media with complex phase functions.

## 4.A Alternative view on scaling-based sampling

Sampling $n$ independent exponentially distributed random numbers and scaling them to unit $L^1$ norm is a method to generate uniform random points on the unit $n$-simplex with surface area $\frac{1}{(n-1)!}$. This is a special case of sampling the Dirichlet distribution with $\alpha = 1$ [Dev86, p. 594]. An alternative view on our sampling algorithm is that we first sample a point on the unit simplex and then project it accordingly. Since the scaling transformation is bijective in this case, this view does not require explicit marginalization.

## 4.B Equivalence to OMNEE

We show that our constraint-based sampling (Section 4.3.3) has the same PDF as OMNEE [HWD22] for the case $n = 2$ and forward-only scattering. Without loss of generality, we can express $O$ using a rotation $\tilde{R}$ and the angle $\theta$ between $\boldsymbol{\omega}'_1$ and $\boldsymbol{\omega}'_2$:

$$O = \tilde{R} \begin{pmatrix} 0 & \sin\theta \\ 0 & 0 \\ 1 & \cos\theta \end{pmatrix}. \tag{4.19}$$

With $a = \sqrt{1 + \cos\theta}$ and $b = \sqrt{1 - \cos\theta}$ its SVD is

$$O = \tilde{R} \begin{pmatrix} b\sqrt{\frac{1}{2}} & a\sqrt{\frac{1}{2}} & 0 \\ 0 & 0 & 1 \\ a\sqrt{\frac{1}{2}} & -b\sqrt{\frac{1}{2}} & 0 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & b \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{pmatrix}. \tag{4.20}$$

The arc length of the feasible region is found as

$$s \arccos(n_1 \cdot n_2) = s \arccos(\cos\theta) = s\,\theta \tag{4.21}$$

using the normal vectors that define its intersected halfspaces

$$n_1 = \frac{1}{z} \begin{pmatrix} a^{-1}\sqrt{\frac{1}{2}} \\ b^{-1}\sqrt{\frac{1}{2}} \end{pmatrix}, \; n_2 = \frac{1}{z} \begin{pmatrix} -a^{-1}\sqrt{\frac{1}{2}} \\ b^{-1}\sqrt{\frac{1}{2}} \end{pmatrix}, \tag{4.22}$$

where $z = \sqrt{\frac{1}{2a^2} + \frac{1}{2b^2}}$. Inserting into Eq. (4.11), we obtain the result from OMNEE

$$\frac{1}{s\,\theta}\, a\, b\, s^2 = \frac{s\sqrt{1 - \cos^2\theta}}{\theta} = \frac{s\sin\theta}{\theta}. \tag{4.23}$$

## 4.C Comparison to the track rotation estimator

The track rotation estimator (TRE) [HDM76] is a closely related estimator from the field of neutron transport. Like our bridge sampling, it also rotates subpaths sampled by a random walk to connect two

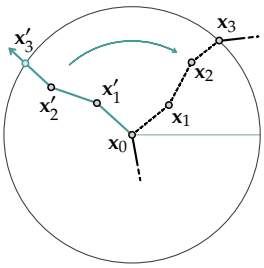



**Figure 4.18:** *To sample a connecting subpath between $\mathbf{x}_0$ and $\mathbf{x}_3$, the track-rotation estimator extends a path starting from $\mathbf{x}_0$ until it intersects a sphere with radius $\|\mathbf{x}_3 - \mathbf{x}_0\|$ centered at $\mathbf{x}_0$. It places a vertex at the intersection point $\mathbf{x}'_3$ and proceeds to rotate the subpath into place, similar to our bridge sampling.*

vertices. In contrast to our approach, the subpath is not scaled, but its number of vertices is sampled as part of the random walk. The TRE came to our attention only after publication of our original algorithm, since it was not well known in the computer graphics community.

In this section, we compare our bridge sampling to the TRE and related estimators. We first summarize the original version of the TRE, before introducing adaptations to our application and combining the TRE with our bridge sampling. This results in several novel estimators, which we evaluate on our test scene from Section 4.5. Overall, our bridge sampling remains the most robust estimator in this comparison, although TRE is equally effective when the number of subpath vertices is unlimited.

### 4.C.1 Modified track rotation estimator

**Original TRE.** The original version [HDM76] constructs connecting subpaths between $\mathbf{x}_0$ and $\mathbf{x}_n$ by beginning an analog random walk at $\mathbf{x}_0$. It makes a connection every time it intersects a sphere with radius $\|\mathbf{x}_n - \mathbf{x}_0\|$ (see Fig. 4.18). The subpath is rotated around $\mathbf{x}_0$ such that its last vertex matches $\mathbf{x}_n$, as in our bridge sampling. Scaling is not necessary, since the distance between its first and last vertex is the desired one by construction.

**Termination on first sphere crossing.** In our application, crossing the sphere multiple times is unlikely, since we target strongly forward-scattering phase functions. It would also be less practical to handle an undetermined number of connecting subpaths instead of just sampling a single one. Therefore, we do not record all sphere intersections, but terminate the random walk at the first intersection. While this results in some energy loss, we did not observe significant bias with the parameters used in our comparison.

**Implicit capture.** In the original TRE, when the random walk terminates after a maximum number of sampled vertices without crossing the sphere, it is considered an absorption event and no connection is formed. When reaching the maximum number of vertices, we thus connect $\mathbf{x}_0$ and $\mathbf{x}_n$ with probability $T(\mathbf{x}'_{n-1}, \mathbf{x}'_n)$, otherwise we estimate zero contribution. We recognize this as a Russian roulette estimator based on the outcome of the very last sampled distance. To reduce variance, we instead form the connection always and multiply with the escape probability, i.e. the transmittance. This yields a lower variance estimator by the Rao-Blackwell theorem.

**Sampling at least one scattering event.** A minor disadvantage of the approach so far is that we would terminate the random walk early when we cross the sphere without any scattering event. We could sample the direct connection between $\mathbf{x}_0$ and $\mathbf{x}_n$ in this case, but it might be desirable to handle direct connections separately. This is the case in our evaluation renderings, where we consider only subpaths with at least one scattering event. We explicitly restrict sampling of the first distance to be inside the sphere and renormalize the PDF accordingly, as done by Villemin et al. [VWF18]. This ensures that we do not sample zero-scattering subpaths in the TRE.

### 4.C.2 Extension of the TRE with bridge sampling

The random walk in the TRE perfectly importance samples all considered terms of the measurement contribution with unbounded steps in theory. However, in practice we have to limit the number of vertices of the subpath and terminate the random walk when we have not crossed the sphere. When we reach the maximum number of vertices, the transmittance of the last vertex to the sphere surface does not cancel with the PDF anymore and becomes a source of variance in the estimator. This is the main weakness of the TRE, as this last transmittance is not importance sampled. Our bridge sampling can handle this outcome better by non-analog scaling of the subpath, which distributes the additional track length more evenly among path segments.

Therefore, we combine both approaches: once we are about to reach the maximum number of vertices, we switch to bridge sampling. We use bridge sampling only for the subpaths with maximum vertex count. Therefore, it becomes unnecessary to sample the number of vertices explicitly. However, we have to account for terminated subpaths in the bridge sampling PDF.

Since TRE terminates its random walk as soon as we intersect the sphere around $\mathbf{x}_0$, any preliminary subpath with vertices outside this sphere would not reach the maximum vertex count. When we marginalize out the scale of the preliminary subpath in the bridge sampling PDF, we thus have to exclude scale factors that would be impossible to sample. We formalize this constraint using the definitions $r = \left\|\mathbf{x}'_n - \mathbf{x}'_0\right\|_2$ and $s = \|\mathbf{x}_n - \mathbf{x}_0\|_2$ and introduce $R$ as the upper bound for $r$:

$$\forall i \in \{1, \dots, n-1\}\colon \frac{r}{s} \leq \frac{s}{\|\mathbf{x}_i - \mathbf{x}_0\|_2} \quad \Leftrightarrow \quad r \leq \frac{s^2}{\max_{i\in\{1,\dots,n-1\}} \|\mathbf{x}_i - \mathbf{x}_0\|_2} =: R. \tag{4.24}$$

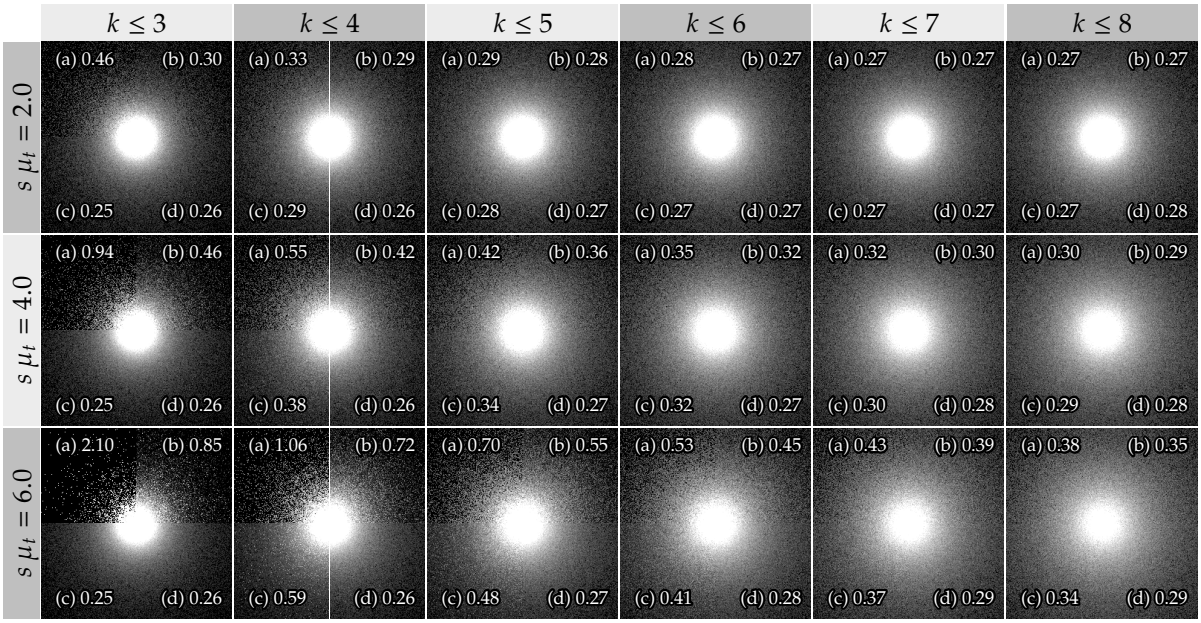
We repeat the derivation in Eq. (4.7) with $R$ as the upper integration limit:

$$\int_0^R p_{\bar{D}'}\left(\frac{r}{s}\bar{d}_{1,n}\right)\frac{r^{n-1}}{s^{n-3}}\,\mathrm{d}r = \frac{\lambda^n}{s^{n-3}}\int_0^R \exp\left(-\frac{\lambda\left\|\bar{d}_{1,n}\right\|_1}{s}r\right) r^{n-1}\,\mathrm{d}r \tag{4.25a}$$

$$= \frac{\lambda^n}{s^{n-3}}\,\gamma\left(n, \frac{R\lambda\left\|\bar{d}_{1,n}\right\|_1}{s}\right)\left(\frac{\lambda\left\|\bar{d}_{1,n}\right\|_1}{s}\right)^{-n} \tag{4.25b}$$

$$= \frac{s^3}{\left\|\bar{d}_{1,n}\right\|_1^n}\,\gamma\left(n, \frac{R\lambda\left\|\bar{d}_{1,n}\right\|_1}{s}\right). \tag{4.25c}$$

Here, $\gamma$ refers to the lower incomplete gamma function. This result is in line with the previous one, since $\lim_{x\to\infty}\gamma(n,x) = \Gamma(n) = (n-1)!$. The PDF in product vertex area measure with $r$ bounded from




(a) TRE RR (b) TRE without RR (c) TRE + Bridges (d) Bridges with second moment

**Figure 4.19:** *Comparison of different TRE-based estimators and our bridge sampling on different extinctions $\mu_t$ and limits on the path length $k$. The inset numbers show the RMSE to a reference. TRE has high variance when it reaches the maximum number of vertices, especially with Russian roulette (a). This can be improved by using bridge sampling for the last vertex (c), although only using bridge sampling (d) still results in lower error. When the limit on $k$ is well above the expected number of collisions, all estimators result in similar error.*

above by $R$ is therefore

$$p_{\bar{X}}(\bar{\mathbf{x}}_{1,n-1}) = \left(\prod_{i=1}^{n} \frac{1}{d_i^2}\right) p_{\bar{\Omega}'}(\bar{\boldsymbol{\omega}}'_{2,n}) \, \frac{s^3 \, \gamma\left(n, R\lambda \left\|\bar{d}_{1,n}\right\|_1 s^{-1}\right)}{\left\|\bar{d}_{1,n}\right\|_1^n}. \tag{4.26}$$

**Alternative combinations.** It would also be possible to combine the TRE and bridge sampling on the last vertex using MIS. We have tried to select the technique using the last preliminary vertex distance. If escaping the sphere, we use the intersection point as the last vertex without scaling as in the TRE. Otherwise, we scale the subpath as with bridge sampling. However, in our testing this MIS combination gave near-identical results to the approach discussed above. Therefore, we chose the simpler combination, where we always use bridge sampling with the last vertex.

### 4.C.3 Results

In Fig. 4.19, we compare our modified TRE with and without Russian roulette (RR), the combination of TRE and bridge sampling, and our bridge sampling estimator, which uses a tabulated second moment to sample the number of vertices (see Section 4.4). All renderings use mean cosine $g = 0.95$ and 16 samples per pixel. The results of the estimators mainly differ when the limit on $k$ is close to the expected number of collisions. Only in this case do we have to terminate without an intersection of the sphere, which increases variance for TRE. As expected, the estimator with Russian roulette is always worse

than the one without. Its combination with bridge sampling further reduces variance, but is not able to consistently reach the low error of the bridge sampling estimator. When the limit on $k$ is sufficiently loose, all estimators have low variance, resulting in mostly equal error. It is possible that TRE will scale better to even longer paths, since our bridge sampling eventually deteriorates with many vertices.

### 4.C.4 Conclusion

The track rotation estimator (TRE) uses a similar idea to our bridge sampling and has similar variance with a loosely bounded number of vertices. However, when its random walk reaches the maximum number of vertices, variance increases. The TRE is limited by its analog free-flight sampling, whereas our bridge sampling handles this case better via scaling of the preliminary subpath. We combined both methods by using bridge sampling when we reach the maximum number of vertices. Although still increasing variance slightly in comparison to bridge sampling only, the combination could be a viable alternative in some cases. Its variance increase is moderate in most cases and it does not require precomputation and table lookups, like our second-moment-based sampling of the number of vertices for bridge sampling. Future work might find better ways to combine TRE and bridge sampling via MIS, which could reduce variance even further.

# 5 Interactive Light Transport Simulation

While the main focus of this dissertation is high-quality offline rendering, interactivity enables a wider range of applications, including preview rendering for scene modeling or data exploration. Advances in graphics processing units (GPUs) allow us to consider algorithms similar to those in offline light transport simulation for interactive applications. Nevertheless, interactivity demands different design decisions, which come mainly from the strict constraint on render times to be on the order of 10–100 ms. This is usually achieved by a combination of highly parallel execution on GPUs, compromises on physical accuracy, and the exploitation of temporal and spatial coherence.

In particular, reusing information in light transport simulation, while crucial for interactive rendering, is also an active research topic in offline rendering. In this chapter, we discuss two different theoretical contributions that enable information reuse in interactive applications. The first one (Section 5.1) is concerned with mixture models with uncountably many, implicit components and the conditions that allow for their sampling. This enables path guiding with limited storage. The second one (Section 5.2) facilitates the combination of different shift mappings in ReSTIR, making the reuse of samples more robust. It was developed in the context of simulating subsurface scattering, which is a specific type of volumetric transport.

## 5.1 Markov chain mixture models for real-time direct illumination

While mixture models have proven to be useful for offline path guiding, storage and computational demands hinder their adoption in interactive applications. Fitting mixture models can be an expensive process and requires sufficient training data that has to be gathered during rendering. Highly dynamic content requires fast temporal adaptation, limiting the utility of distributing computation among multiple frames.

In "Markov Chain Mixture Models for Real-Time Direct Illumination" (MCMM) [DSH*23], we represent a mixture model implicitly by a stochastic process. This allows for lightweight fitting and storage, as well as fast adaptation to changes in illumination. Formally, we define a continuous mixture model with components $V(\cdot \mid t)$ and parameters $t \in \mathcal{T}$:

$$p(x) = \int_{\mathcal{T}} p(t)\, V(x \mid t)\, \mathrm{d}t. \tag{5.1}$$

Here, $p(t)$ is the component density, which corresponds to the weight in a discrete model. We construct a stochastic process with state space $\mathcal{T}$ and interpret the current state $t$ as a random sample of our mixture model. Perhaps surprisingly, we can use it to sample $x \sim V(x \mid t)$ and make unbiased estimates without knowing $p(t)$, employing a *stochastic MIS* (SMIS) estimator [WGGH20]. This leaves considerable flexibility for designing the component sampling process.

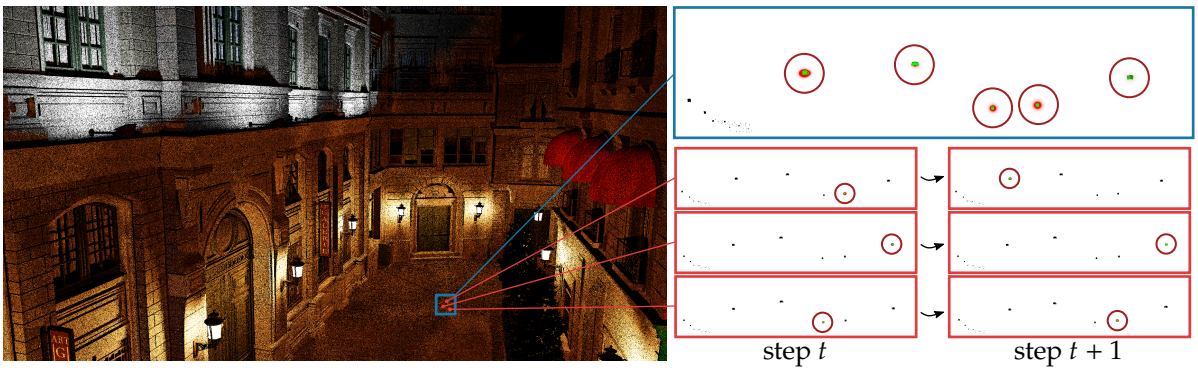



**Figure 5.1:** *Left: Direct illumination from many small light sources rendered using MCMM with 8 samples per pixel in 10.8 ms. Right: The sampled mixture model components visualized using latitude / longitude coordinates. The red insets show the vMF distributions (red) held by individual pixels at consecutive time steps. Light sources are shown in black and their overlap with vMF distributions in green. We encircle the vMF distributions for better visibility (dark red). The blue inset shows the mixture that emerges when combining the individual components, i.e. from nearby pixels and multiple time steps. Figure reproduced from [DSH*23] with permission.*

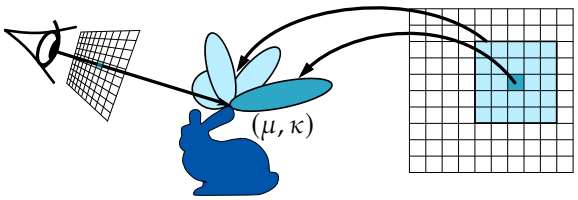



**Figure 5.2:** *Each Markov chain state defines the parameters $(\mu, \kappa)$ of a single vMF distribution (dark blue). Since each pixel contains one Markov chain state, a sample of the full, continuous mixture model can either be obtained by collecting vMF components over multiple time steps, or in a window around the center pixel (light blue). Figure reproduced from [DSH*23] with permission.*

We use this construct in MCMM for guiding the sampling of direct illumination without explicit information about light sources (see Fig. 5.1). Every pixel holds the state of one Markov chain, each one defining the parameters of a von Mises-Fisher (vMF) distribution. We interpret this as a sample from a continuous mixture model and use it to importance sample a direction toward a light source (see Fig. 5.2). The Markov chain transition is informed by the result of evaluating this direction sample. Resampling states from neighboring pixels spreads information about successful vMF distributions. This overall approach was later extended to indirect illumination [AHD25].

In the remainder of this section, we extend the theory of SMIS estimation in order to prove correctness of the application in MCMM, as well as to clarify capabilities and limitations in general.

### 5.1.1 Stochastic multiple importance sampling with dependent samples

Following West et al. [WGGH20], we define the $n$-sample SMIS estimator for $I = \int_X f(x)\,\mathrm{d}x$:

$$\langle I \rangle_{\mathrm{SMIS}} = \sum_{i=1}^{n} \frac{\dot{w}_i(X_i)\, f(X_i)}{p(X_i \mid T_1, \ldots, T_n)}, \tag{5.2}$$

with random *techniques* $T_i$ and *samples* $X_i$ with joint PDF $p(T_1, \dots, T_n, X_1, \dots, X_n)$. In our definition, we replaced $p(X_i \mid T_i)$ with the more general $p(X_i \mid T_1, \dots, T_n)$, which is equivalent in simple cases. The weighting heuristic $\dot{w}$ can be any valid MIS weighting heuristic [VG95b], i.e. it has to satisfy the following conditions:

**(W1)** $\sum_{i=1}^{n} \dot{w}_i(x) = 1$ when $f(x) \neq 0$,

**(W2)** $\dot{w}_i(X_i) = 0$ when $p_i(X_i \mid T_1, \dots, T_n) = 0$.

We adopt the proof by West et al. to show that $\langle I \rangle_{\text{SMIS}}$ is indeed an unbiased estimate of $I$:

$$\mathbb{E}\left[\langle I \rangle_{\text{SMIS}}\right] = \mathbb{E}_{(T_1,X_1),\dots,(T_n,X_n)}\left[\langle I \rangle_{\text{SMIS}}\right] \tag{5.3a}$$

$$= \mathbb{E}_{T_1,\dots,T_n}\left[\mathbb{E}_{X_1,\dots,X_n}\left[\langle I \rangle_{\text{SMIS}} \mid T_1, \dots, T_n\right]\right] \tag{5.3b}$$

$$= \mathbb{E}_{T_1,\dots,T_n}\left[\mathbb{E}_{X_1,\dots,X_n}\left[\sum_{i=1}^{n} \frac{\dot{w}_i(X_i)\, f(X_i)}{p(X_i \mid T_1, \dots, T_n)} \,\middle|\, T_1, \dots, T_n\right]\right] \tag{5.3c}$$

$$= \mathbb{E}_{T_1,\dots,T_n}\left[\sum_{i=1}^{n} \mathbb{E}_{X_i}\left[\frac{\dot{w}_i(X_i)\, f(X_i)}{p(X_i \mid T_1, \dots, T_n)} \,\middle|\, T_1, \dots, T_n\right]\right] \tag{5.3d}$$

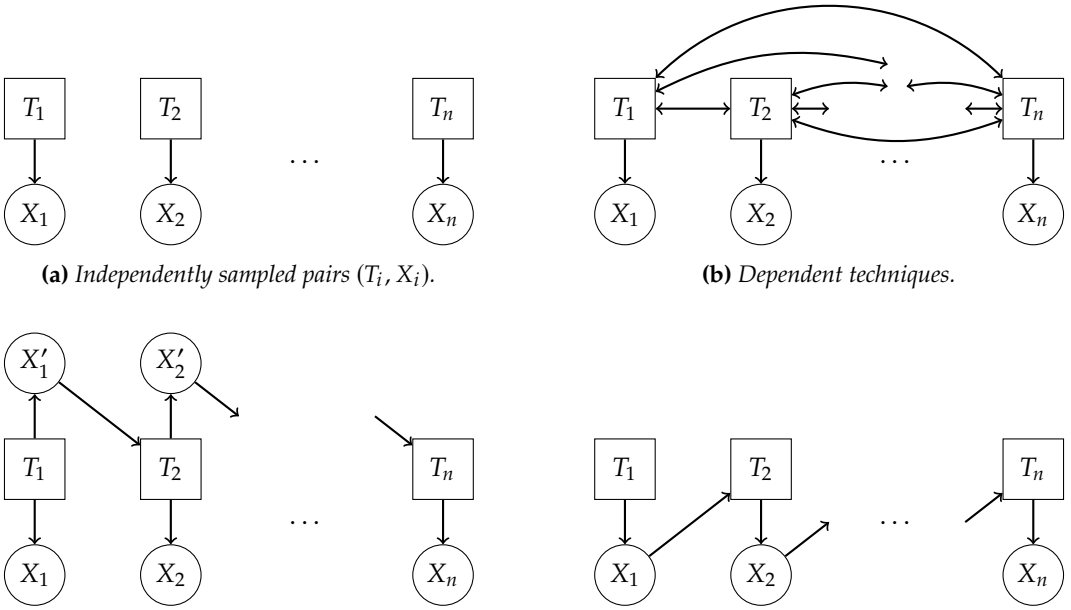
$$= \mathbb{E}_{T_1,\dots,T_n}\left[\sum_{i=1}^{n} \int_{\mathcal{X}} \frac{\dot{w}_i(x)\, f(x)}{\cancel{p(x \mid T_1, \dots, T_n)}} \cancel{p(x \mid T_1, \dots, T_n)}\, \mathrm{d}x\right] \tag{5.3e}$$

$$= \mathbb{E}_{T_1,\dots,T_n}\left[\int_{\mathcal{X}} \underbrace{\sum_{i=1}^{n} \dot{w}_i(x)}_{\overset{\text{(W1)}}{=} 1} f(x)\, \mathrm{d}x\right] = \mathbb{E}\left[\underbrace{\int_{\mathcal{X}} f(x)\, \mathrm{d}x}_{=I}\right] = I. \tag{5.3f}$$

The main insight here is that we can sample a set of MIS techniques randomly from a continuous space, without needing to know the probability of that selection. Given an appropriate weighting function $\dot{w}(x)$ that respects (W1) and (W2), the conditional expectation of the estimator given $T_1, \dots, T_n$ is already equal to the integral $I$. The outer expectation does not contain any random variables, making it already equal to $I$, independently of $p(T_1, \dots, T_n)$. In the following, we discuss some examples of different sampling schemes to highlight capabilities and limitations of the SMIS estimator.

**Independently sampled pairs** $(T_i, X_i)$ **(Fig. 5.3a).** This is the assumption made by West et al. for their proof, even though it is not strictly necessary. In this case, the PDFs can be simplified. Since $X_i$ is conditionally independent of $T_{j\neq i}$ given $T_i$, $p(X_i \mid T_i) = p(X_i \mid T_1, \dots, T_n)$. The same is true for the cases shown in Figs. 5.3b and 5.3c.

**Dependent techniques (Fig. 5.3b).** In this example, techniques are not sampled independently. For the proof above, we take the expectation given all random techniques. Therefore, inside this conditional expectation, dependencies between any techniques do not matter. Since we did not break any assumptions of the proof above, this example still results in an unbiased SMIS estimator. For MCMM, we make use of this insight for resampling techniques from the previous frame. Since we are resampling from a fixed set, the resulting techniques are correlated, but this does not compromise




**(a)** *Independently sampled pairs* $(T_i, X_i)$.

**(b)** *Dependent techniques.*

**(c)** *Dependent techniques through extra samples.*

**(d)** *Techniques dependent on previous samples.*

**Figure 5.3:** *Examples of different possible sampling schemes for use with the SMIS estimator.*

unbiasedness. Another application could be to sample a set of independent points first, and then produce techniques through combination, e.g. use the convex hull of a subset of points.

**Dependent techniques through extra samples (Fig. 5.3c).** Here, an additional sample $X_i'$ is used to select the technique for the next sample $T_{i+1}$, which leads to a dependency $T_i \to T_{i+1}$. Since $X_i'$ is not used for the estimate itself, this is a special case of the previous example and no bias is introduced. The algorithm by Martino et al. [MELC17] uses this scheme for adapting importance sampling to the integrand similarly to what we do in MCMM. However, we save the cost of the extra sample by choosing $T_i$ dependent on $X_i$ and avoid the dependency by permuting techniques between multiple estimators running in parallel.

**Techniques dependent on previous samples (Fig. 5.3d).** In this case, we would use a sample $X_i$ directly to choose the next $T_{i+1}$. Although this scheme would be desirable for adaptive importance sampling, we explicitly avoid this case as explained in the previous example. While this scheme would be supported with the SMIS estimator, it is difficult to apply it in practice. This is because it is usually intractable to compute the required conditional PDF $p(X_i \mid T_i, T_{i+1})$, since we only know $p(X_i \mid T_i)$ and $p(T_{i+1} \mid X_i)$:

$$p(X_i \mid T_i, T_{i+1}) = \frac{p(X_i, T_i, T_{i+1})}{p(T_i, T_{i+1})} = \frac{p(T_{i+1} \mid X_i)\, p(X_i \mid T_i)\, \cancel{p(T_i)}}{p(T_{i+1} \mid T_i)\, \cancel{p(T_i)}} \tag{5.4}$$

For the missing term $p(T_{i+1} \mid T_i)$, we would have to marginalize out $X_i$. Since it appears in the denominator in Eq. (5.4) and therefore in the numerator of Eq. (5.2), we could replace it with an uncorrelated estimate, which however would need at least one additional sample, like the previous example did. Second, one needs to be careful about proper weighting functions, since the reversed PDF

(Eq. (5.4)) could be zero for some $x$, even where $p(X_i = x \mid T_i) \neq 0$. Because of (W2), this makes it more difficult to cover the full domain according to (W1).

### 5.1.2 Summary

We have outlined conditions for the applicability of the SMIS estimator with different kinds of dependencies between techniques and samples. In MCMM, these dependencies are mostly avoided explicitly by clever permutation of Markov chain states of neighboring pixels. Correlations are only introduced by resampling neighboring states from the previous frame. As shown above, this still does not affect correctness of the approach. We hope that our observations can provide insights and aid in the design of similar methods in the future.

## 5.2 ReSTIR subsurface scattering for real-time path tracing

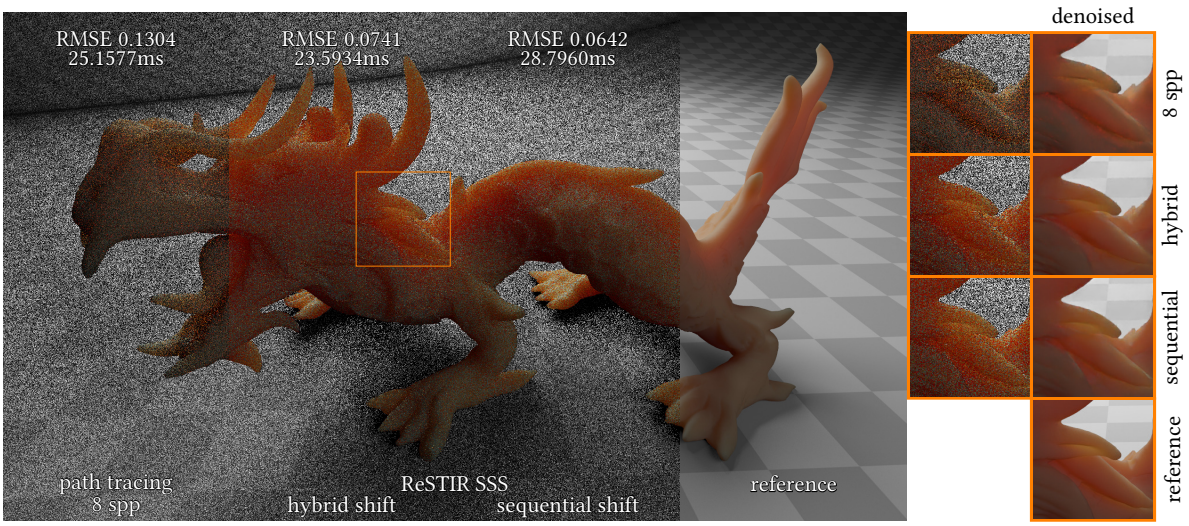



**Figure 5.4:** *The material on the dragon model shows characteristic blurring and translucency effects of subsurface scattering, which also lead to additional noise. Applying ReSTIR with the proposed shift mappings to the simulation of subsurface scattering reduces noise and denoising artifacts, in particular in regions with prominent scattering. Figure reproduced from [WSD24] with permission.*

The term subsurface scattering describes a specific kind of volumetric light transport that occurs in dense media bounded by surface geometry. While its appearance is close to diffuse surface reflection, volumetric scattering introduces characteristic subtle blurring and translucency of thin geometry (see Fig. 5.4). This is important for the realistic appearance of certain materials, most prominently skin on characters.

Because of these properties, there exist specialized solutions for computing subsurface scattering. In high-quality offline rendering, subsurface scattering is simulated as full volumetric light transport, but using specialized importance sampling techniques [Kd14] or nested lightweight integrators [BAC*18; CFS*18]. More approximate solutions augment surface transport by replacing the BSDF with a *bidirectional surface scattering distribution function* (BSSRDF), which models transport between separate

entry and exit points. Models for the BSSRDF use simplifying geometric assumptions to precompute scattering, traditionally based on diffusion theory [JMLH01] and as a function of distance between entry and exit point. The BSSRDF can be used to blur rendered images in a postprocess, but this approach is limited by its missing access to information about hidden surfaces. It can also be used with path tracing by sampling exit points on surfaces [KKCF13]. This, however, leads to additional noise, which is problematic for interactive applications.

In "ReSTIR Subsurface Scattering for Real-Time Path Tracing" (ReSTIR-SSS) [WSD24], we apply reservoir-based spatio-temporal importance resampling (ReSTIR) [BWP*20] to the rendering of subsurface scattering with BSSRDF importance sampling. ReSTIR reduces noise by enabling unbiased reuse of samples in space and time with constant storage. While it has been applied to path tracing before as ReSTIR PT [LKB*22], we extend this approach to support BSSRDFs.

### 5.2.1 Combining shift mappings

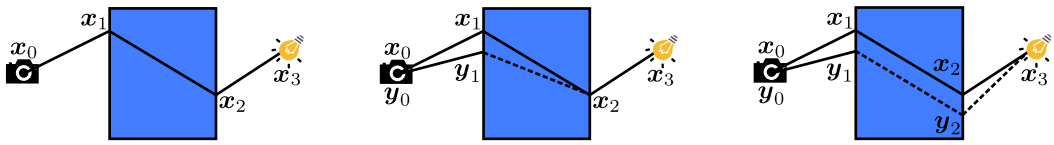
Shift mappings bijectively map a base path to a similar offset path in a different domain. While originally designed for gradient-domain rendering [LKL*13], shift mappings allow unbiased reuse of paths between different domains in ReSTIR [LKB*22], e.g. neighboring pixels. As there is no unique notion of "similarity" of paths, various shift mappings have been proposed [HGP*19; LKB*22]. It is difficult to design robust shift mappings, or to choose the most suitable one for a given base path [HGP*19, Sec. 8.1].

We adopt two basic techniques for constructing offset paths: copying vertices of the base path and reconnecting to it as soon as possible (reconnection), or copying random numbers of the base path (random replay). In ReSTIR-SSS, we found different configurations to benefit from two shift mappings when shifting paths with a subsurface interaction (see Fig. 5.5): we either reconnect to the exit point (reconnection shift), or use random replay to sample a new exit point and reconnect afterwards (delayed reconnection shift).

ReSTIR PT also delays reconnection to a "connectable" vertex, which it deterministically classifies based on surface roughness and path vertex distance (hybrid shift). We introduce a similar heuristic based on distance and orientation for BSSRDF reconnection. It is effective in image regions where one of the options is clearly superior. However, this approach can fail in regions where the criterion gives less consistent results, leading to visible correlation artifacts. Therefore, we also introduce a second approach, where we apply reconnection and delayed reconnection in alternation (sequential shift).

This is however not straightforward, since these shifts operate in primary sample space or path space for different parts of a path. Through shifting we can create paths for which the corresponding random numbers are unknown, but required for subsequent random replay. We can often obtain random numbers by inverting the sampling process, but this is not always possible. For example, ambiguities occur when we choose between multiple techniques with overlapping domains during sampling. This is the case in ReSTIR-SSS, since we sample the exit point conditional on a chosen projection axis and color channel.

Mapping paths from path space to primary sample space has been investigated in the context of Metropolis light transport [Pan17; OKH*17; BJNJ17]. A problem similar to ours occurs when combining path space and primary sample space mutations. The theoretical foundation for a solution is to extend path space with extra dimensions to match the dimensionality of primary sample space. This allows us to define a bijective mapping. In practice, the extra dimensions can be randomly sampled before




**(a)** *Subsurface path* **(b)** *Reconnection shift* **(c)** *Delayed reconnection shift*

**Figure 5.5:** *Scattering inside the blue object is approximated using a BSSRDF model and represented by a path with entry and exit points (a). ReSTIR-SSS uses a combination of two shift mappings: either we reconnect to the base path through the object (b), or delay reconnection by reusing random numbers of the base path (c). Figure reproduced from [WSD24] with permission.*

inversion. This approach also has application in path guiding [HSRM25, Ch. 2, Sec. 6]. In this section, we first explain the theoretical extension of path space and illustrate this with an example, before applying probabilistic inversion to ReSTIR-SSS.

### 5.2.2 Path space extension

Ambiguities in the mapping from path space $\mathcal{P}$ to primary sample space $\mathcal{U} = [0,1)^*$ originate in a mismatch of their dimensionality. For example, to sample a mixture model, we would use a random number to select one component to sample. This random number has no equivalent in path space. Therefore, going in reverse, we cannot in general recover the component index from the sample, let alone the random number that was used to select it.

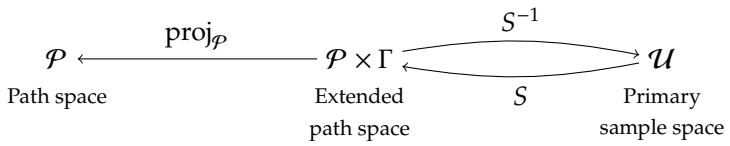



**Figure 5.6:** *Sampling only requires a mapping from primary sample space $\mathcal{U}$ to path space $\mathcal{P}$, which is not necessarily injective. We recover invertibility by extending $\mathcal{P}$ with additional dimensions $\Gamma$, such that a bijective mapping $S : \mathcal{U} \to \mathcal{P} \times \Gamma$ can be defined.*

We can solve this problem by extending path space with extra dimensions $\Gamma = [0,1)^*$, such that the dimensionalities of $\mathcal{P} \times \Gamma$ and $\mathcal{U}$ match. This allows us to define a bijection $S$ between both spaces (see Fig. 5.6). Integration over these dimensions does not change the value of the integral, as they do not occur inside the measurement contribution function.

$$\int_{\mathcal{P}} f(\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} = \int_{\mathcal{P}} \underbrace{\int_{\Gamma} \mathrm{d}\bar{\boldsymbol{\gamma}}}_{=1} f(\bar{\mathbf{x}})\,\mathrm{d}\bar{\mathbf{x}} = \int_{\mathcal{P}\times\Gamma} f(\bar{\mathbf{x}})\,\mathrm{d}(\bar{\mathbf{x}}, \bar{\boldsymbol{\gamma}}) \tag{5.5}$$

In the following, we provide an example for the case of sampling a mixture model and show how to define the mapping from $\mathcal{U}$ to $\Gamma$. Since the PDF is formally defined by the Jacobian determinant of $S^{-1}$, we choose $S$ such that its Jacobian remains consistent with the original PDF.

#### 5.2.2.1 Example: bijective sampling of a mixture model

In our example, we use a mixture of $T$ techniques $S_t$ for importance sampling $x$:

$$S\colon \mathcal{U} \to \mathcal{P} \times \Gamma \qquad (u_1, u_2) \mapsto (x, \gamma) =: (S_{t(u_1)}(u_2), \gamma(u_1)), \tag{5.6}$$

For each technique $t$, we define its PDF $p(x \mid t)$ and weight $\alpha_t \coloneqq P(t) > 0$, with $\sum_{j=1}^{T} \alpha_j = 1$. Then, the PDF of $x$ and the probability of $t$ given $x$ are

$$p(x) = \sum_{j=1}^{T} \alpha_j p(x \mid t = j), \qquad P(t \mid x) = \frac{\alpha_t p(x \mid t)}{p(x)}. \tag{5.7}$$

**Sampling $S$.** We first select a technique by index $t \sim P(t) = \alpha_t$ and then use $S_t$ to sample $x$:

$$\text{select the unique } t \text{ such that } \sum_{j=1}^{t-1} \alpha_j \le u_1 < \sum_{j=1}^{t} \alpha_j, \qquad x = S_t(u_2). \tag{5.8}$$

Next, we need to define the mapping from $u_1$ to $\gamma$, such that

1. we are able to invert $S$, i.e. uniquely infer $t$ and $u_1$ from $\gamma$,
2. the Jacobian determinant of $S$ matches the reciprocal PDF: $|\det J_S(\bar{\mathbf{u}})| \stackrel{!}{=} \frac{1}{p(x)}$.

The following definition fulfills both of these requirements:

$$\gamma = \frac{u_1 - \sum_{j=1}^{t-1} \alpha_j}{\alpha_t} P(t \mid x) + \sum_{j=1}^{t-1} P(t = j \mid x). \tag{5.9}$$

Inversion is possible, since $\gamma$ is inside an interval unique to $t$ due to our choice of $t$ in Eq. (5.8)

$$u_1 \in \left[\sum_{j=1}^{t-1} \alpha_j, \sum_{j=1}^{t} \alpha_j\right) \Rightarrow \frac{u_1 - \sum_{j=1}^{t-1} \alpha_j}{\alpha_t} \in [0, 1) \Rightarrow \gamma \in \left[\sum_{j=1}^{t-1} P(t = j \mid x), \sum_{j=1}^{t} P(t = j \mid x)\right), \tag{5.10}$$

and the Jacobian determinant of $S$ is indeed equal to the reciprocal PDF of $x$

$$|\det J_S(u_1, u_2)| = \left|\det \begin{pmatrix} 0 & \frac{\partial x}{\partial u_2} \\ \frac{\partial \gamma}{\partial u_1} & \frac{\partial \gamma}{\partial u_2} \end{pmatrix}\right| = \left|\frac{\partial x}{\partial u_2}\right| \left|\frac{\partial \gamma}{\partial u_1}\right| = \left|\frac{1}{p(x \mid t)}\right| \left|\frac{P(t \mid x)}{\alpha_t}\right| = \frac{1}{p(x)}. \tag{5.11}$$

**Inversion $S^{-1}$.** We use Eq. (5.10) to determine the technique index $t$ from $\gamma$:

$$\text{select the unique } t \text{ such that } \sum_{j=1}^{t-1} P(t = j \mid x) \le \gamma < \sum_{j=1}^{t} P(t = j \mid x). \tag{5.12}$$

Then, we invert Eq. (5.9) to get $u_1$, and use $S_t^{-1}$ to get $u_2$ from $x$:

$$u_1 = \frac{\gamma - \sum_{j=1}^{t-1} P(t = j \mid x)}{P(t \mid x)} \alpha_t + \sum_{j=1}^{t-1} \alpha_j, \qquad u_2 = S_t^{-1}(x). \tag{5.13}$$

As expected, the Jacobian determinant of this mapping is equal to the PDF of $x$:

$$|\det J_{S^{-1}}(x, \gamma)| = \left|\det \begin{pmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial \gamma} \\ \frac{\partial u_2}{\partial x} & 0 \end{pmatrix}\right| = \left|\frac{\partial u_2}{\partial x}\right| \left|\frac{\partial u_1}{\partial \gamma}\right| = \left|p(x \mid t)\right| \left|\frac{\alpha_t}{P(t \mid x)}\right| = p(x). \tag{5.14}$$

### 5.2.3 Probabilistic inversion in ReSTIR

We are now able to bijectively map paths between the extended path space and primary sample space. To use shift mappings in both path space and primary sample space, we assume integration in (extended) path space without loss of generality and embed primary sample space mappings in path space. Given a shift mapping $\tilde{T}\colon \mathcal{U} \to \mathcal{U}$, we can embed it as $T = S \circ \tilde{T} \circ S^{-1}$ to get a mapping $T\colon \mathcal{P} \times \Gamma \to \mathcal{P} \times \Gamma$. For example, when $\tilde{T}$ is the identity function to implement random replay, the Jacobian of $T$ is the ratio of the PDF in original and target domain if the mapping is constructed as in the previous section.

Unfortunately, we still do not know the extra dimensions $\bar{\boldsymbol{\gamma}} \in \Gamma$ of a path that was constructed by reconnection and not through sampling. In MLT, it is possible to randomly sample these as part of a mutation [Pan17; OKH*17; BJNJ17]. We also sample a uniform random $\bar{\boldsymbol{\gamma}} \in \Gamma$ before applying $S^{-1}$, but the validity of this approach with ReSTIR requires further justification.

Since shift mappings are required to be deterministic, we cannot make a probabilistic inversion part of the shift itself. It is, however, possible to apply Metropolis-Hastings mutations between resampling in ReSTIR [SLK*24], when adjusting the unbiased contribution weight by the ratio of the target function evaluated before and after mutation. Uniform sampling of $\bar{\boldsymbol{\gamma}}$ has unit PDF and does not change the value of the target function. Therefore, it can be considered a Metropolis-Hastings mutation with acceptance probability equal to 1, i.e. we always accept a proposed $\bar{\boldsymbol{\gamma}}$. Since we do not require $\bar{\boldsymbol{\gamma}}$ immediately, but only in the next resampling stage, we can lazily sample it between resampling as a mutation. For correctness, it is still necessary to keep $\bar{\boldsymbol{\gamma}}$ the same for each path during one resampling stage.

### 5.2.4 Summary

With this construction, it is possible to use probabilistic inversion in ReSTIR without introducing bias. This allows the combination of shift mappings defined in path space and primary sample space, as long as we can derive an inverse mapping from extended path space to primary sample space similar to $S^{-1}$ above. In ReSTIR-SSS, this frees us from the a priori choice of a shift mapping for each path. Instead, we alternate both shift mappings, resulting in reduced correlation artifacts.

## 5.3 Conclusion

In this chapter, we have discussed two approaches to making light transport simulation more efficient in interactive applications. These are similar in that they both consider a stochastic subset of sampling

techniques from a continuous space. However, while in MCMM the techniques are continuous distributions that facilitate exploration, in ReSTIR-based approaches techniques are fixed subpaths. The latter allows for improved temporal accumulation in return, while MCMM has issues with representing too many light sources at once. Both demonstrate that in the domain of interactive rendering, it can be worthwhile to explore the frontiers of well-known mathematical frameworks in order to drive the design of novel algorithms and to adapt them to the tradeoffs dictated by GPU hardware.

# 6 Selective Caching in Procedural Texture Graphs for Path Tracing

In the preceding chapters, we discussed variance reduction as a prerequisite for the practical convergence of Monte Carlo light transport simulation. Another key factor to consider is the runtime cost of constructing paths and evaluating their contribution, as this limits the number of samples that remain feasible. The material evaluation at each path vertex constitutes a significant share of the overall cost, especially in large-scale rendering. Complex materials are driven by detailed textures, i.e. high-resolution inputs to BSDFs that are mapped to surfaces.

*Procedural* texturing is essential in modern 3D content generation for providing a controllable, resolution-independent approach to texture generation. Tools like Blender [Ble25] and Adobe Substance 3D [Ado25] use procedural texture graphs (PTGs), where nodes define texture generators or filters, and edges determine information flow. By combining procedural primitives (e.g. noise functions) with operations like gradient mapping and distortion, users can efficiently create diverse, intricate textures. Adjusting node parameters reduces manual effort while achieving rich visual detail.

In this chapter, we address the efficient implementation of procedural texturing into a rendering system based on path tracing. We show that performance is greatly affected by the strategy used for evaluating a PTG, e.g. precomputation, on-demand evaluation or caching. Furthermore, the ideal choice of strategy depends on the application context. We therefore present a new method to dynamically select which texture graph nodes to cache within a rendering system in order to minimize runtime cost.



**Figure 6.1:** *The PartyTug scene with procedural textures, which we evaluate using our framework integrated into a path tracer. For each material in this scene, we instantiate one randomly selected procedural texture graph. Through caching, our approach reduces total render time for 256 samples per pixel to 38% of the time taken by rendering using shade-on-hit, i.e. evaluating each texture footprint individually.*

## 6.1 Introduction

When rendering procedurally textured scenes, the task of evaluating many PTGs can become a substantial part of the total rendering work. Optimization is especially challenging in Monte Carlo-based rendering, e.g. path tracing, since the renderer requires random access to all textures. Procedural texture graphs provide a highly compact, generative representation and offer flexibility in how they are implemented: a rendering system can decide which parts of the texture to generate and when; this offers potential for more efficient resource management, balancing memory usage, performance, and visual fidelity.

A straightforward strategy is to precompute all output textures at high resolution. This allows us to fully decouple procedural texturing from rendering. Since textures are evaluated only once, multiple accesses to the same texture regions share results of a PTG evaluation. On the other hand, many texture regions will not be required in high resolution, due to partial occlusion, distance to the camera, or blur induced by camera lenses or a light scattering process. Since information about this is not available in advance, upfront evaluation potentially computes many unused results. In addition, precomputation increases latency and can require a significant amount of intermediate storage space.

Graph-based shading systems like Open Shading Language (OSL) [GSKC10] take the opposite approach by delaying graph evaluation until render time. This allows us to evaluate graphs only on actual texture accesses by the renderer. Potential benefits include the reduction of computation and storage overhead, as well as balancing the use of memory and compute resources. However, as different texture accesses in OSL evaluate the PTG individually, they do not share evaluation results, thus impeding the overall efficiency. Tile-based caching can sometimes offset this drawback. However, it involves upfront work that has to amortize for it to increase efficiency, similar to full precomputation.

There is no universal answer to the question of which of these approaches is preferable in a given configuration. Therefore, our goal is to avoid a fixed choice in advance and leverage the full flexibility of the procedural representation instead. In this chapter, we introduce a novel framework for efficiently evaluating procedural texture graphs in a path tracer: We delay the choice of the evaluation *strategy* to the rendering process itself. This allows us to decide whether to cache based on information about the rendered scene in a fine-grained way, at the level of individual parts of a PTG. Our framework also provides filtered PTG evaluation and prediction of computational costs to guide optimization. We review the state of the art in the next section, before giving a more detailed overview of our framework.

## 6.2 Background and related work

Keeping up with increasing shading and texturing complexity is a recurring challenge in production rendering [CCC87; FHL*18; CFS*18; BAC*18; KCSG18; GIF*18]. Rendering systems typically expose powerful programmable interfaces to artists for controlling shading, through shader languages or shading graphs [HL90; GSKC10]. These allow us to combine many layers of textures and procedurally defined patterns and noises. Executing many user-provided shader programs can become a bottleneck for rendering. Accessing textures involves a great deal of I/O, while procedural functions can be computationally intensive.

With the shift toward path tracing [KFF*15], this has become more difficult. In the typical shade-on-hit architecture, a shader has to be executed at every path vertex interaction. The incoherent nature

of random paths leads to nonlocal memory access and divergent execution between different paths, hindering the use of SIMD instructions. Manuka [FHL*18] is designed around improving shading efficiency by fully decoupling shader execution and path tracing (shade before hit). Another approach is to try finding coherence in a larger batch of paths through sorting [ENSB13; ÁBWM16; LGXT17; CFS*18], sometimes with hardware support [NVI22; Int22]. In large and complex scenes, there usually remains at least some level of incoherence.

**Texture caching.** Software caches were introduced early on [Pea90] to improve random texture access, with their overall design mostly standing the test of time [Gri08; CFS*18]. As the cache is discrete while texture space is continuous, accessing a texture cache involves reconstruction. The effectiveness of texture caching relies on the assumption that the required level of detail for reconstruction is often much lower than the original texture resolution. Peachey [Pea90] describes this as the "principle of texture thrift", which states that the amount of texture information used for rendering should be limited by the output resolution, as any additional detail would be imperceptible. This ensures that the working set of textures remains bounded and, most importantly, independent of the original texture resolution, allowing it to fit efficiently into the cache. However, this assumption holds only if lower-resolution versions of the texture can be efficiently stored and reconstructed, which inherently links caching to prefiltering. In practice, this is commonly achieved using resolution pyramids [Wil83] (mipmaps).

**Footprints.** Reconstruction bandlimit is usually represented as the linear transformation of a circular kernel, derived from a first-order approximation of the light transport process. This results in an elliptical *footprint*, which we can use for efficient filtering, e.g. using EWA [GH86; Hec89]. There exist various theories for computing appropriate footprints in ray and path tracing, including ray [Ige99] and path differentials [SW01], conical approximation [Ama84; ANA*19] or based on frequency analysis [BYRN17] or half-vector distributions [KHD14; HKD15]. Accurate footprints are generally difficult to obtain, since unidirectional methods neglect the (unknown) frequency content of either illumination or camera importance [BYRN17]. Bidirectional footprints are only available once a complete path is formed, presenting a challenge to incremental path construction. Due to discontinuities, in particular visibility, even a complete path does not carry the required information [ZD20] and aggregation would be necessary. Therefore, in practice, footprints are rather approximate and often overestimated in order to favor caching.

**Filtering with nonlinearities.** Linear prefiltering is valid only for values that occur linearly in the rendering integral. Nonlinearities require more advanced methods [BN12]. This presents a challenge for caching in a procedural texture graph, as caching and thus filtering might occur before a nonlinear node. Note that in shading graphs, this problem is often ignored in practice to facilitate efficient caching using mipmaps [GSKC10]. Heitz et al. [HNPN13; HNPN14] and Grenier et al. [GSDT22] filter color maps applied to Gaussian noises. Fournier and Sauvage [FS24] show that normal distributions are relevant approximations in the context of nonlinear texture transitions. Yang et al. [YB18] use normal distributions to automatically transform shader programs to bandlimited variants using genetic search and additional approximate filtering rules [DBLW15]. We also represent cached results as normal distributions in order to apply nonlinear operations to them.

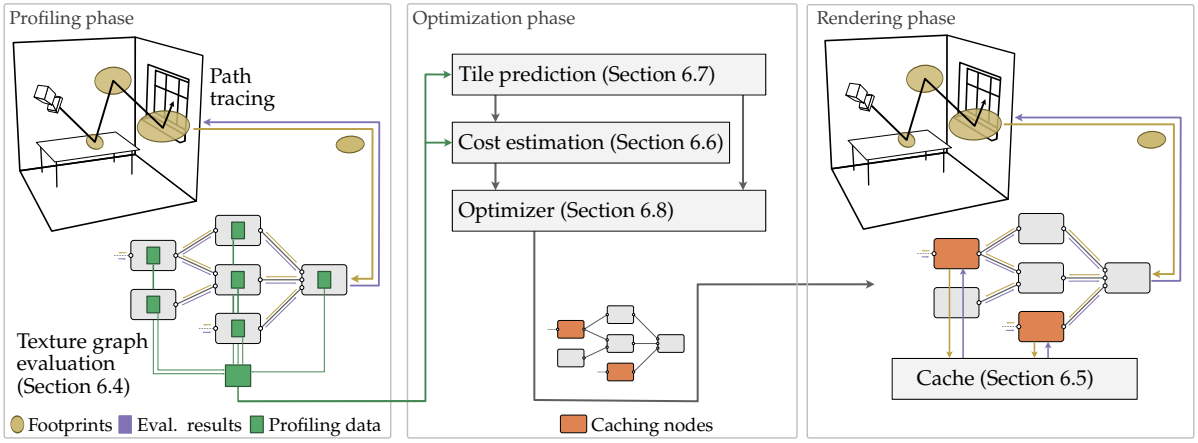



**Figure 6.2:** *Overview of our system. The algorithm is structured in three phases. First, during the profiling phase, path tracing generates texture footprints which are collected and used to query the texture graph with direct (uncached) evaluation, to collect profiling data. Then, an optimization is performed to determine the most cost-effective position of cache nodes in the graph. Finally, the rendering phase continues tracing paths and querying the texture graph for footprints, using the optimized cache locations. See the indicated sections for more details.*

**Procedural texture graphs.** Procedural texture graphs (PTGs) are directed acyclic graphs that describe a process for synthesizing textures. Source nodes produce parametric patterns or noise textures [LLC*10], while inner nodes process and combine their inputs, e.g. through color transformations, blending or warping. Nodes in a PTG can access arbitrary pixels of their inputs, making them more powerful than common shading graphs [GSKC10]. PTGs thus present a very generic framework, in which many procedural techniques can be implemented as custom nodes. Recent works explore the inverse problem of obtaining graph parameters [SLH*20] or topologies [GHS*22] or both [LWS*25] from an example image, while we focus on improving the integration of PTGs into rendering and their efficient evaluation.

**Caching procedurals.** Reiner et al. [RLD*12] cache the evaluation of costly implicit functions to accelerate rendering on GPU hardware. Similarly, Fujieda et al. [FH22] cache the output of shading graphs. They always cache the largest subgraph that only depends on texture coordinates, in order to maximize work saved by reusing cached values. We instead try to maximize efficiency by explicit search for nodes to cache and in addition consider filtering of nonlinear results.

## 6.3 Overview

Our work revolves around maximizing shading performance by balancing different computational strategies. In particular, these are on-demand and upfront evaluation of footprints via tile-based caching. Evaluating each footprint individually when requested by the renderer limits shading work to what is necessary, but misses the potential of sharing results between individual evaluations. Caching can leverage these redundancies, but necessitates resampling. This leads to upfront evaluation of grids of footprints, which has to amortize before any efficiency is gained. Available memory capacity and

bandwidth further limit efficient reuse of shading results.

Which of the two strategies is more efficient depends on many parameters of the system and its use case, e.g. the cost of operations, available memory and Monte Carlo sample count. Further, the choice between the two options also depends on properties of the rendered scene. Shape and distribution of footprints are different for each material, since it depends on camera position and (indirect) visibility. The structure of the texture graph can make caching outputs of certain nodes preferable to others. A global choice for one of the strategies might therefore limit overall efficiency. Our goal is to leverage this unused potential by making more fine-grained choices to optimize efficiency for any specific use case individually.

In the following, we describe our framework to make these choices as the result of an optimization that is driven by data we collect from inside the rendering system in a profiling phase. We show an overview of the different phases and components in Fig. 6.2.

In a path tracer, we compute elliptical footprints. Our method does not rely on a particular definition of these footprints, though we document our derivation in Appendix 6.A to facilitate reproducibility. We evaluate texture graphs on footprints using approximate filtering (Section 6.4). This enables prefiltering inside the texture graph, which is a prerequisite for efficient caching at inner nodes. As a second option for graph evaluation, we introduce our texture cache in Section 6.5. Its design is similar to a traditional tile cache commonly used in rendering systems. We discuss tradeoffs of using caching, in particular possibly insufficient amortization and storage space requirements. These motivate a data-driven decision of whether to enable caching for each node.
This decision is inherently a global optimization problem. On one hand due to a global space constraint, which might require to prioritize caching of nodes that benefit the most. On the other hand, caching certain nodes reduces the value of caching others, e.g. there would be redundancy in caching a linear chain of nodes. In the end, we will formulate an ILP problem whose solution is the optimal selection of nodes to cache (Section 6.8). This ILP problem requires several input parameters from the rendering system, namely evaluation cost and space requirements of different evaluation strategies. For this, we introduce a parametric measure of evaluation cost in Section 6.6. We describe how we estimate cost for different strategies for a target sample count, based on data collected in the profiling phase. While the cost without caching is roughly linear in sample count and simple to extrapolate, we also require the total number of tiles that will be requested by the rendering system. This is needed to assess the cost of caching as well as its storage impact. To this end, we introduce a statistical model to predict the number of unique texture tiles requested at the target sample count (Section 6.7).

## 6.4 Texture graph evaluation with filtering

In our work, we assume each material in the scene to be defined by a procedural texture graph (PTG) with a single output node. To evaluate a PTG for a given footprint requested by the path tracer, we evaluate the footprint at the output node of the PTG. Each node can then recursively evaluate its input nodes as necessary, possibly with transformed or duplicated footprints.

**Prefiltering motivation.** In our PTG, nodes are queried with footprints and return solutions to a filtering integral (see Eq. (6.1)). We choose this approach over alternatives, such as stochastic evaluation on

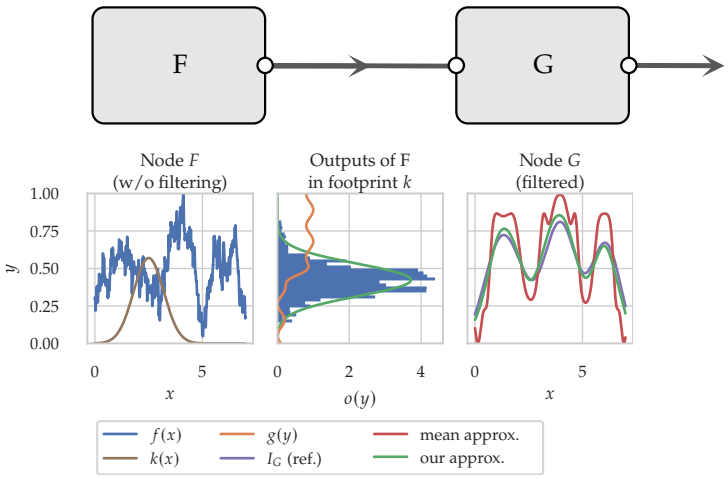



**Figure 6.3:** *We interpret the filter kernel $k(x)$ as a probability density of a random variable $X$ and define $Y := f(X)$ with density $o(y)$. Evaluating a footprint to $o$, we can apply the nonlinear function $g(y)$ afterwards and still get correct results. In practice, we approximate $o$ using a normal distribution.*

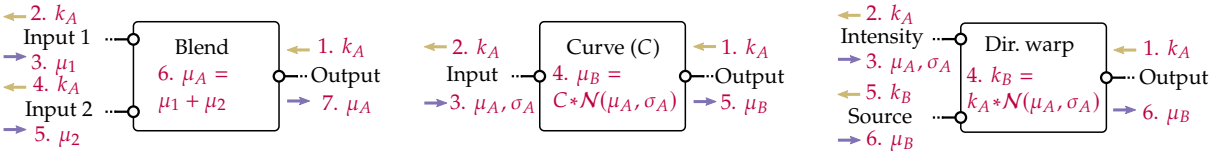
sampled points, for multiple reasons. While stochastic evaluation can sometimes amortize additionally introduced noise by reducing computation and memory traffic [PWSF24], nested integrals as they occur in PTGs are much more challenging to handle [WM24]. Further, for effective texture caching it is essential that sparsity of requested evaluation points is compensated by lower cache resolution. This is usually achieved by resolution pyramids and the size of footprints being roughly proportional to their spacing. Consequently, we need to carry footprint information through the graph to support caching inside of it, or for using mipmapping for image source nodes, or for leveraging built-in filtering of procedural models.

**Filtering nonlinearities.** Even though we use prefiltering inside the graph, our goal is to preserve the appearance of filtering at the output only. Results of footprint evaluation need to allow for the application of subsequent nonlinear operations as if applying them *before* prefiltering. As we show below, the distribution of values inside a footprint is a suitable representation of intermediate results inside the graph, while its mean would be insufficient. In practice, we approximate the full distribution using normal distributions, similar to previous work [HNPN13; HNPN14; GSDT22; YB18]. We assume that our graph output represents a value occurring linearly in the shading integral (e.g. albedo), although similar techniques are commonly used to filter normal maps [DHI*13].

### 6.4.1 Filtering theory

In the following, we show how the distribution of values allows us to apply subsequent nonlinear operations to an evaluated footprint. Consider the node $F$ in the graph in Fig. 6.3 that represents a function $f$ on texture coordinates $x \in \Omega$. Its filtered output is the inner product of $f$ with a footprint $k$, which is usually some kernel function:

$$I_F = \int_\Omega f(x)\, k(x)\, \mathrm{d}x. \tag{6.1}$$




**(a)** *Additive blending is linear, thus we can evaluate the means of $k_A$ for both inputs separately.*

**(b)** *For evaluating a nonlinear transfer function $C$, we convolve a Gaussian approximation of the input in $k_A$ with $C$.*

**(c)** *For directional warping, the "Intensity" input controls the strength of the warping applied to "Source". We use the Gaussian approximation to compute a new footprint $k_B$.*

**Figure 6.4:** *Three different nodes evaluating the mean value $\mu_A$ for a footprint $k_A$. The numbers signify the order of execution. For instance, the Blend node receives a request for $k_A$ (1) and passes this on to Input 1 (2), obtains $\mu_1$ in return (3), repeats the same with Input 2 (4, 5), computes the output mean (6), before finally passing it on to the Output (7).*

We take a probabilistic view and interpret $k$ as the probability density function of a random variable $X$. Following Eq. (6.1), filtering $F$ is equivalent to the expected value $\mathbb{E}[f(X)] = I_F$.

Returning to the more general problem, we want to compute an intermediate result from $F$ that we can use to filter the downstream node $G$, which applies a function $g$ pointwise to the outputs of $F$. Its filtered output is

$$I_G = \mathbb{E}[g(f(X))] \neq g(\mathbb{E}[f(X)]) = g(I_F), \tag{6.2}$$

which shows that we cannot trivially use $I_F$ as an intermediate result. To compute $I_G$, we need to capture more information about the distribution of $f(X) =: Y$ (see Fig. 6.3), which its PDF $o_F$ achieves:

$$I_G = \mathbb{E}\left[g(f(X))\right] = \mathbb{E}[g(Y)] = \int_{\mathbb{R}} g(y)\, o_F(y)\, \mathrm{d}y. \tag{6.3}$$

Therefore, we use the PDF $o_F$ as the result of footprint evaluation.

### 6.4.2 Filtering approximation

Conceptually, nodes in our graph are queried with a footprint PDF $k$ and return an output PDF $o$. We coarsely approximate both as normal distributions to make this approach efficient. In practice, only pairs of mean and covariance are exchanged between nodes. We show how some typical PTG nodes can be implemented in this framework in Fig. 6.4.

**Footprint PDF $k$ approximation.** We represent footprints as 2D normal distributions like in EWA filtering [GH86]. With this, linear transformation and convolution are simple to implement, which is useful when nodes query their inputs on a modified $k$ (see Fig. 6.4).

**Output PDF $o$ approximation.** We approximate the output distribution of nodes as a normal distribution as well. We evaluate Eq. (6.3) on the fly for its first and second moment, unless there is a more specialized implementation available for an operation. Other approximations like the unscented transform [Uhl95] might be a viable alternative for some operations. For multi-channel output, we treat each channel independently.

**Filtering accuracy.** In actual texture graphs, footprints and node outputs are not exactly normally distributed at all scales and node inputs and color channels can correlate, reducing the accuracy of our approach. We show results of our approximation in Fig. 6.5 and timings and errors in Table 6.1. We evaluate the output image for $1024 \times 1024$ pixels with one footprint per pixel. We compare to a point-sampled reference with 256 samples per pixel, as well as a naive filtering method (mean), which uses just the mean of $o$ as the output of nodes. Mean filtering shows some obvious patterned artifacts, while our approximation produces plausible results with consistently lower error. Error tends to increase with wider footprints, as nodes with multiple inputs get no information about their correlation inside the footprint area. We also compare to reconstruction from the cache (+ cache), which Section 6.5 describes in detail. To reconstruct a footprint, the cache will require smaller footprints to be evaluated than the original one. This generally reduces the error slightly compared to direct evaluation without the cache, but increases the time to evaluate the base resolution ($l = 10$). As spacing of evaluated footprints in the cache depends on their radius, the cache effectively decouples the density of evaluated footprints from the output resolution. Therefore, caching is significantly faster for coarse filtering ($l = 7, 5$) evaluated at high resolution, i.e. if the density of footprints to reconstruct is high in relation to their size.

**Discussion.** In principle, we could make the filtering error arbitrarily small by closer approximation of the involved PDF, or supersampling. However, coarse results of filtering are often accepted in practice, i.e. through mipmaps themselves or nonlinear computations on linearly filtered values in shading graphs. While improving accuracy is an interesting avenue for future work, we believe our approximation serves as a basic model for investigating efficient graph evaluation and caching.

## 6.5 Texture cache

To reuse results of previously evaluated footprints, we introduce a texture cache into our framework. It follows a conventional design by managing texture tiles in a mipmap-like pyramidal structure. Each tile stores an $8 \times 8$ grid of evaluated isotropic footprints. In level $i$ of the resolution pyramid, the footprint radius is $0.5^{i+1}$ and the spacing is $0.5^i$ in texture space, i.e. a greater level index corresponds to a finer level. Organization into tiles helps to amortize cache management and improves locality of operations. We assume a global cache, where evaluated results of any node from any texture graph can be stored. In summary, tiles are indexed by texture graph and node ID, tile position and resolution level.

### 6.5.1 Footprint evaluation strategies

The introduction of the cache enables different options for evaluating a footprint at any node.

**Direct evaluation.** As before, we can evaluate a footprint by executing the operation defined by a node, while evaluating all required footprints from input nodes recursively.

**Caching evaluation.** If using caching at a node, we use EWA filtering to reconstruct arbitrary elliptical footprints from the cache of isotropic footprints. This usually involves interpolation between a coarse and a fine resolution level. We compute texture tiles on demand that are missing in the cache, but

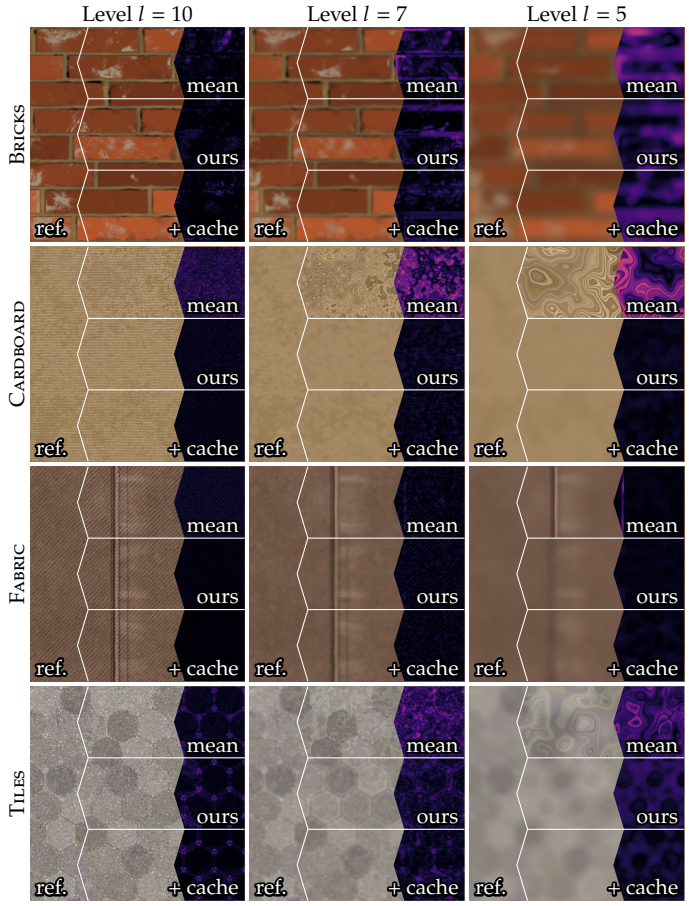



**Figure 6.5:** *Our Gaussian filtering approximation with footprint radius 0.5px ($l = 10$), 4px ($l = 7$) and 16px ($l = 5$) in comparison to naive filtering (mean). We overlay a ꟻLIP error map to a reference on the right. Our filtering preserves appearance better than using just the mean, which shows obvious patterned artifacts. Reconstructing our approximation from cache (+ cache) often improves results, since it evaluates the graph using smaller footprints.*

**Table 6.1:** *Timing and mean ꟻLIP errors for the comparison in Fig. 6.5. We list per sample timings for the point sampled references (*).*

| | **time (s)** | | | **error (ꟻLIP $\cdot 10^{-3}$)** | | |
|---|---|---|---|---|---|---|
| Level $l$ | $l = 10$ | $l = 7$ | $l = 5$ | $l = 10$ | $l = 7$ | $l = 5$ |
| BRICKS | *13.0 | *12.4 | *13.1 | | | |
| mean | 1.8 | 1.5 | 1.2 | 41.6 | 87.2 | 170.9 |
| ours | 4.9 | 4.6 | 4.1 | 27.5 | 81.2 | 154.0 |
| + cache | 26.0 | 0.7 | 0.2 | 26.9 | 49.9 | 114.7 |
| CARDBOARD | *6.5 | *6.6 | *6.2 | | | |
| mean | 3.7 | 3.4 | 3.2 | 129.9 | 237.9 | 301.9 |
| ours | 11.5 | 10.0 | 9.7 | 47.6 | 59.1 | 51.6 |
| + cache | 61.0 | 1.7 | 0.3 | 39.6 | 52.4 | 50.4 |
| FABRIC | *1.4 | *1.4 | *1.5 | | | |
| mean | 2.6 | 2.6 | 2.6 | 71.0 | 54.7 | 51.9 |
| ours | 7.4 | 7.6 | 7.7 | 30.5 | 50.0 | 43.5 |
| + cache | 38.9 | 1.4 | 0.3 | 19.7 | 43.7 | 40.4 |
| TILES | *46.4 | *47.0 | *47.1 | | | |
| mean | 2.0 | 1.8 | 1.6 | 70.2 | 173.1 | 179.6 |
| ours | 7.3 | 6.9 | 6.7 | 61.0 | 94.8 | 123.0 |
| + cache | 38.1 | 1.2 | 0.3 | 46.5 | 78.5 | 77.6 |

required to evaluate a footprint. For this, we use direct evaluation of the node on the footprint grid defined by the missing tile.

### 6.5.2 Choosing a strategy

For any footprint that we evaluate at a node, we now have two options for computing its result.Since there is not one universal strategy that is the most efficient one in all situations, we have to consider carefully which of the strategies fits best.

**Amortization.** Even though caching enables reuse of computed results, without sufficient redundancy in requested tiles it might not amortize its overhead. For caching, we generally have to reconstruct a footprint from multiple footprints of finer resolution in order to keep high frequencies. Therefore, the cost of reconstruction is inherently higher at first, and caching only pays off when sufficient reuse occurs. Evaluating isotropic footprints to reconstruct from is thus an investment of additional upfront computation for a future reduction in computation.

**Available space.** Another consideration for the efficiency of caching evaluation is the available space for storing results. If we would have to exceed available space because of newly evaluated tiles, we have to either evict old tiles or drop new ones. Both options make amortization less likely, as they can significantly reduce the period of potential reuse of a tile. Therefore, in cases where tiles would need to be evicted from the cache, it is usually more beneficial to never place them in the cache in the first place and instead use direct evaluation. In our work, we thus try to avoid eviction by using caching only when we will not exceed the available storage space. Once a tile has been computed, we thus assume it to stay in the cache and not be evicted.

**Data-driven caching decision.** Since all of these considerations depend on the specific distribution of footprints at hand, our goal is to make a data-driven decision about caching as part of the rendering system. We decide which evaluation strategy to use for each node of each PTG. While in principle a more fine-grained decision would be possible, e.g. treating regions of the texture differently, this adds additional complexity for reconstruction and for finding suitable texture partitions.

## 6.6 Measure for computational cost

As input to our optimization, we need to assess how efficient different strategies are at evaluating a texture graph on a set of footprints. To this end, we introduce a parametric model as a measure for the computational cost. This acts as a very simplified simulator: it allows us to compare different options without actually running all of them. It also allows us to reason about sensitivity of the outcome on different properties of hardware or rendering systems. Measured wall clock time can be incorporated into our model by setting parameter values according to timings.

### 6.6.1 Cost components and coefficients

The basic unit of our model is the local cost of evaluating a node with coefficients $\Theta = (c, \lambda_*)$ on a footprint $x$:

$$\text{cost}(x \mid \Theta) = c \cdot \left(\lambda_{\text{const}} + \lambda_{\text{iso}} \cdot \text{anisotropy}(x) + \lambda_{\text{none}} \cdot \text{size}(x)\right). \tag{6.4}$$

We use the same equation for modeling the cost of reconstruction from cache with a global set of coefficients $\Theta_{\text{cache}}$. It consists of the following components:

- $c$ is a node-specific constant, e.g. measured time per operation. This could depend on parameters of the node and on the type of its output (gray or color).
- anisotropy$(x)$ is proportional to footprint anisotropy. This models the cost of evaluation for a node for isotropic prefiltering, and simulates the cost of footprint assembly for anisotropic queries.
- size$(x)$ is proportional to footprint area. This component models the cost of evaluation, if no efficient prefiltering is available for a node and simulates point sampling in base resolution.
- $\lambda_{\text{const}}, \lambda_{\text{iso}}, \lambda_{\text{none}}$ are node-specific weights to blend between the different filtering models, i.e. constant prefiltering, isotropic prefiltering, or no prefiltering. We constrain these to $\sum \lambda_* = 1$.

### 6.6.2 Aggregate cost

The local cost from the previous section does not include the cost of evaluating input nodes. In addition, for modeling the cost of cached evaluation, we need to consider a set of footprints at once. We now extend our definitions accordingly.

We begin by defining the set of input footprints as

$$\text{In}_i(x) = \{(j, y) \mid \text{evaluating } x \text{ at node } i \text{ depends on evaluating } y \text{ at node } j\}. \tag{6.5}$$

This is useful to express the total cost of the direct evaluation of a footprint $x$ at node $i$, including the evaluation of inputs, recursively:

$$\text{cost}_\Sigma(x \mid \Theta_i) = \text{cost}(x \mid \Theta_i) + \sum_{(j,y)\in \text{In}_i(x)} \text{cost}_\Sigma(y \mid \Theta_j). \tag{6.6}$$

Next, let $\mathcal{F} = (F, w)$ be a weighted set of footprints, i.e. $F$ is a set of footprints and $w\colon F \to \mathbb{R}^+$ is a weight function. The (total) cost for direct evaluation of $\mathcal{F}$ is then defined by

$$\text{cost}_*^{\text{direct}}(\mathcal{F} \mid \Theta) = \sum_{x\in F} w(x)\ \text{cost}_*(x \mid \Theta). \tag{6.7}$$

To model caching, we additionally introduce the function $T$

$$T(\mathcal{F}) = \{t \mid t \text{ is a tile required for reconstructing any } x \in F\}, \tag{6.8}$$

and the function $Z$ to map a set of tiles to their associated footprints

$$Z(\mathcal{T}) = (\{x \mid x \text{ is a footprint in tile } t, t \in \mathcal{T}\}, \mathbf{1}). \tag{6.9}$$

We can now define the total cost of caching the evaluation of $\mathcal{F}$ as

$$\text{cost}_{\Sigma}^{\text{cache}}(\mathcal{F} \mid \Theta) = \underbrace{\text{cost}^{\text{direct}}(\mathcal{F} \mid \Theta_{\text{cache}})}_{\text{reconstruction cost}} + \underbrace{\text{cost}_{\Sigma}^{\text{direct}}(Z(T(\mathcal{F})) \mid \Theta)}_{\text{cache fill cost}}. \tag{6.10}$$

### 6.6.3 Computing the value per node in practice

In order to compare different evaluation strategies for a node, we require a representative set of footprints, on which we can evaluate our cost model. We use a profiling phase to record such a set, i.e. we render few samples per pixel using direct evaluation and store the set of requested footprints at each node.

**Histogram representation.** To represent these recorded sets compactly, we assume that the footprint queries are somewhat uniformly distributed for a node. In particular, we assume there is no correlation between footprint shape and a particular tile in the cache. We store the footprints in a 2D histogram at each node. Each bin is indexed by the $\log_2$ length of a footprint's major and minor axis. This information about a footprint is sufficient to evaluate all terms in our local cost model, i.e. its size and anisotropy. This histogram representation is approximate due to discretization.

**Cache tiles.** In addition, we record the set of required cache tiles $T(\mathcal{F})$ at each node. The number of unique tiles per level is sufficient information for evaluating the cost model for caching evaluation, as well as for estimating the required cache space.

**Propagating footprint sets.** For the purpose of computing the total cost of a node $i$, i.e. including every evaluated upstream (i.e. transitive input) node $j$, in general the footprint set at $j$ will differ from the recorded set. This is because the recorded set includes every evaluation of $j$, even the ones not caused by an evaluation of $i$. Second, caching node $i$ would change the footprint set at $i$ from $\mathcal{F}$ to $Z(T(\mathcal{F}))$. While we can easily simulate this change at $i$ using the recorded set of tiles, it is less obvious how it affects the footprint set at upstream nodes.
Therefore, we introduce an approximation to the function In (see Eq. (6.5)), which we also record using the footprints we evaluated during profiling. It allows us to simulate the distribution of footprints at immediate input nodes, given the requested footprint distribution at a node. In this way, we propagate the recorded distribution to all upstream nodes and use it to evaluate the cost model. For approximation, we again rely on discretization and store a 2D histogram of changes between incoming and outgoing footprints of a node. We store separate histograms for each input node $j$ to a node $i$, and for each cache level at $i$, i.e. the $\log_2$ minor axis length of incoming footprints at $i$. Each histogram is normalized to the number of requested footprints at $i$. The histogram bins represent the $\log_2$ change in footprint axis length between the footprints requested at $i$ and $j$. We use these to transform a distribution of footprint requests from $i$ to the one that is requested from $j$.

**Cost measure evaluation accuracy.** We compare our practical evaluation to the cost evaluated on the actual footprints in Fig. 6.6. We use all combinations of our graphs and scenes (see Section 6.9) and show the distribution of relative error, separated by the different cost terms. For cache fill cost to be accurate, our approximation needs to be able to simulate a change in the footprint distribution. For both

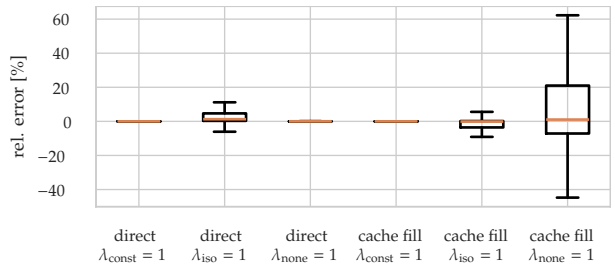



**Figure 6.6:** *Distribution of relative error of our practical cost evaluation. While we achieve low error for most cost terms, simulated cache fill cost is not fully reliable when $\lambda_{none} \neq 0$.*

direct and cache fill cost, constant cost, i.e. footprint count at each node, is predicted with almost no error, while footprint anisotropy ($\lambda_{\text{iso}} = 1$) has a low error. The error of approximate footprint size has high variance for the cache fill cost, while direct cost is always accurate. We presume this is because deviations from the true footprint size have a quadratic influence on the cost. Consequently, we cannot fully rely on predicted cost for $\lambda_{\text{none}} \neq 0$, so we leave $\lambda_{\text{none}} = 0$ for almost all of our results.

## 6.7 Prediction of active texture area

For optimizing the choice of evaluation strategy, our goal is to estimate the cost at a target sample count, while observing only few profiling samples. While for direct evaluation the cost is roughly linear and can be scaled by sample count accordingly, caching behaves differently. In the beginning many newly requested tiles have to be evaluated, leading to a high upfront cost. Once the cache is filled sufficiently, this cost declines and only the usually much lower reconstruction cost remains.

Ultimately, we would like to keep the profiling phase as short as possible to make full use of potential optimization gains. Therefore, the following section is concerned with predicting the progression of unique requested texture tiles at any sample count, based on data from the profiling phase. This is not only important for predicting the cost of caching for a higher sample count, but also for predicting the required cache space. The overall idea we follow is to use a Poisson process to model texture tile requests.

### 6.7.1 Idea

Let $A_n \coloneqq T(\mathcal{F}_n)$ be the set of active texture tiles at sample count $n$, i.e. the ones that the renderer requested at sample count $n$. Our goal is to predict the number of unique tiles $\mathbb{E}[|A_n|]$, given statistics obtained from $k < n$ initial samples about $A_k$. For now, we assume that each tile $t \in A_\infty$ will be requested with uniform probability. This assumption is reasonable, since mipmap pyramids are designed to achieve this approximately. We assume further that tile requests are accurately described by a Poisson process with rate parameter $\lambda$, i.e. the frequency by which each tile is requested. Since each request is independent in the Poisson process, frequency is equal to probability for selecting any specific tile. The number of tiles in the limit will be the reciprocal of this probability, since we assumed it to be uniform:

$$\lambda = \frac{1}{|A_\infty|} \quad \Rightarrow \quad |A_\infty| = \frac{1}{\lambda}. \tag{6.11}$$

The Poisson assumption also allows us to predict the expected number of requested tiles at sample count $n$

$$\begin{aligned}\mathbb{E}\left[|A_n|\right] &= \mathbb{P}\left[\text{request count} > 0 \text{ at sample count } n\right] |A_\infty| \\ &= \left(1 - \exp(-\lambda\, n)\right) |A_\infty|.\end{aligned} \tag{6.12}$$

This reduces our problem to inferring the underlying Poisson rate of the tile request process.

In principle, we found the idea as described above to work well, even though tile requests are not truly independent, since a footprint evaluation can lead to multiple requested tiles. There are however some important considerations to take into account for making this approach practical, which we address in the following.

### 6.7.2 Zero-truncated Poisson mixture model

Actual data reveals significant inhomogeneity of the Poisson process, i.e. request probability is not actually uniform. This can be explained by e.g. occlusion, where tiles are only partially visible. In practice, we solve this by using a mixture model, i.e. we assume a finite number of different Poisson rates. Further, since we only observe tiles that were requested at least once, our average tile count will be biased toward higher values. We correct this by modeling the average tile count using a zero-truncated Poisson distribution.

**Definition.** The Poisson distribution has the probability

$$\mathrm{Pois}(k \mid \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{6.13}$$

which we use to define its zero-truncated version by

$$\mathrm{Pois}_+(k \mid \lambda) = \frac{\mathrm{Pois}(k \mid \lambda)}{1 - \mathrm{Pois}(0 \mid \lambda)} = \frac{\mathrm{Pois}(k \mid \lambda)}{1 - \exp(-\lambda)}. \tag{6.14}$$

Together with the weights $\pi$, we define the $M$-component zero-truncated Poisson mixture, with which we model request counts:

$$p(k) = \sum_{j=1}^{M} \pi_j \, \mathrm{Pois}_+(k \mid \lambda_j). \tag{6.15}$$

### 6.7.3 Model parameter estimation

**Data.** During the profiling phase, for each requested footprint at a node we compute which tiles would be requested and increment a per-tile counter for each request. Afterward, we compute a histogram of all observed request counts $r$. Given $N$ different observed requests counts, the histogram is a sequence of tuples $(r_i, o_i)_{1 \le i \le N}$, where $o_i$ is the occurrence count, i.e. how many tiles were requested $r_i$ times.

**Maximum likelihood estimate.** To obtain the maximum likelihood estimate of a single zero-truncated Poisson component given a mean count $\bar{x}$, we solve the following equation in terms of the Lambert $W$

function:

$$\frac{\lambda}{1-e^{-\lambda}} = \bar{x} \quad \Rightarrow \quad \lambda = \bar{x} + W(-\bar{x}e^{-\bar{x}}) \tag{6.16}$$

**Expectation maximization (EM).** We use the EM algorithm [DLR77] to compute a maximum likelihood estimate of our mixture model. In each iteration of the algorithm, it computes an assignment $a_{i,j}$ for each data point $i$ to component $j$ as the expectation under the current mixture:

$$a_{i,j} = \frac{\pi_j \operatorname{Pois}_+(r_i \mid \lambda_j)}{p(r_i)} \tag{6.17}$$

Then it maximizes the likelihood of component parameters, conditional on the assignments:

$$\pi_j = \frac{\sum_{i=1}^N a_{i,j}\, o_i}{\sum_{i=1}^N o_i}, \quad \bar{x}_j = \frac{\sum_{i=1}^N a_{i,j}\, o_i\, r_i}{\sum_{i=1}^N a_{i,j}\, o_i}. \tag{6.18}$$

We obtain $\lambda_j$ as in Eq. (6.16).

**Initial mixture.** To initialize the EM iteration, we begin with a dense initial mixture, i.e. we choose $M$ on the order of $N$, e.g. $M = 0.25\,N$. To choose each initial $\lambda_j$, we sample from the empirical data distribution $D$, i.e. the discrete distribution with $\Pr(R = r_i) \propto o_i$, and apply a uniform jitter to obtain an initial mean $\bar{x}_j$:

$$\begin{aligned} &R \sim D, \quad J \sim \operatorname{Uniform}(-0.5, 0.5), \\ &\bar{x}_j = \begin{cases} \max\{R + |J|, 1.01\} & \text{if } R = 1, \\ R + J & \text{otherwise.} \end{cases} \end{aligned} \tag{6.19}$$

Since the initial means follow the data distribution, we set $\pi_j = \frac{1}{M}$.

**Merging.** To make EM converge more quickly, we remove redundant components by merging after few initial iterations. We merge greedily and proceed in ascending order of $\lambda_j$. We merge two components with index $j$ and $j+1$ if they are mutually contained within a tolerance $\tau$ of their standard deviations $\sqrt{\lambda}$, i.e. if

$$\lambda_j \geq \lambda_{j+1} - \tau\sqrt{\lambda_{j+1}} \text{ and } \lambda_{j+1} \leq \lambda_j + \tau\sqrt{\lambda_j} \tag{6.20}$$

Merging can usually reduce the number of components greatly. Afterwards, we proceed with EM iterations until convergence.

### 6.7.4 Prediction

We now extend our idea for predicting $\mathbb{E}[|A_n|]$ to our mixture model. First, the inferred Poisson rates $\lambda_j$ are with respect to the number of observations made and need to be normalized for Eqs. (6.11) and (6.12) to be applicable.

For estimating $|A_\infty|$, we normalize $\lambda_j$ to the total number of tile requests $t = \sum_{i=1}^N o_i\, r_i$. We divide

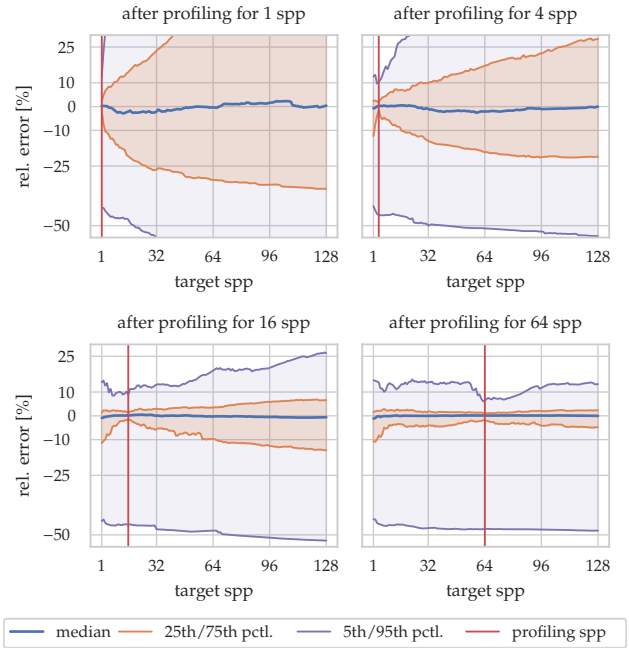



**Figure 6.7:** *Distribution of relative error of our tile number prediction. While the prediction is accurate on average and produces low error in most cases when taking sufficient profiling samples, some outliers remain.*

these between components using the assignments from EM (Eq. (6.17)):

$$t_j = \sum_{i=1}^{N} a_{i,j}\, o_i\, r_i. \tag{6.21}$$

Our estimate for the total number of unique tiles of component $j$ is

$$|A_\infty^j| = \frac{t_j}{\lambda_j}. \tag{6.22}$$

We use the number of footprints at target sample count $\#\mathcal{F}_n$ and at profiling sample count $\#\mathcal{F}_k$ for estimating the expected number of requested tiles, since we expect $\#\mathcal{F}$ to grow linearly:

$$\mathbb{E}\left[|A_n|\right] = \sum_{j=1}^{M} \mathbb{E}\left[|A_n^j|\right] = \sum_{j=1}^{M} \frac{t_j}{\lambda_j} \left(1 - \exp\left(-\underbrace{\frac{\#\mathcal{F}_n\, \lambda_j}{\#\mathcal{F}_k}}_{=\frac{n\lambda_j}{k}}\right)\right). \tag{6.23}$$

**Conservative estimation.** In absence of sufficient data, our approach will underestimate $|A_n|$. This is because mixture components with very low $\lambda$ are unlikely to be observed. To prevent this systematic bias, we add a single component to our mixture to represent unobserved tiles, such that it has negligible influence on the prediction if enough data was observed. We extend our mixture to $M' = M + 1$ components, use a fixed $\lambda_{M'} = 0.01$, and then choose $t_{M'}$, such that when we plug it into Eq. (6.22),

$|A_\infty^{M'}|$ equals 10% of the total estimated unique tiles:

$$\left|A_\infty^{M'}\right| = 0.1 \sum_{j=1}^{M} \left|A_\infty^{j}\right| \quad \Rightarrow \quad t_{M'} = \left(0.1 \sum_{j=1}^{M} \left|A_\infty^{j}\right|\right) \lambda_{M'}. \tag{6.24}$$

Then we use Eq. (6.23) as before, with $M$ switched out for $M'$. Since $\lambda_{M'}$ is set very low, it will only influence the prediction if $k \ll n$.

**Accuracy.** We show results of our prediction in Fig. 6.7, with varying samples per pixel (spp) for the profiling phase. We compare our predicted tile count to actual tile count in all our scenes (see Section 6.9) and show the distribution of relative error. On average, our prediction is accurate even with few profiling samples, although with high variance. Taking more profiling samples generally reduces variance and leads to a low error for most predictions. However, some outliers still remain, even for longer profiling phases. One possible explanation for this behavior could be that we predict the expected value of the number of tiles. Therefore, the error is not purely deviation of the ground truth from our estimate, but also includes the variance of the underlying stochastic process itself. While improving the robustness of this component of our framework is interesting for future work, we believe most error to be low enough for it to be useful in practice.

## 6.8 Optimization of caching decision

With the results from previous section, we can estimate how caching a node would affect overall efficiency. We are, however, still missing an algorithm for turning this information into a decision about which nodes to cache. Our task is to solve an optimization problem, where we want to maximize overall efficiency by picking nodes to cache where this increases efficiency, while staying below a threshold for cache capacity. This is essentially the classic knapsack problem, but we will have to add one further restriction by considering only one stage of caching. In the following, we explain this restriction and the inputs to our optimization, as well as an integer linear programming (ILP) formulation. To obtain a solution, we use a general ILP solver without much overhead.

**Restriction to single-stage caching.** We support only one stage of caching, i.e. we do not consider caching inputs to cached nodes. More formally, there can be at most one caching node on every path from any source node to the graph output (see Fig. 6.8), since multiple stages of caches are often unnecessarily redundant.

### 6.8.1 Integer linear programming formulation

In order to share the cache capacity between all graphs in a scene, we perform a single global optimization after the profiling phase. We use the data gathered during the profiling phase to compute costs and predict tile counts.

**Variables.** For each node $i$ of every graph, we define its weight $w_i$, i.e. its required cache space, using the expected number of tiles $\mathbb{E}[|A_n|]$ (see Section 6.7), its number of output channels $C_i$ and the number of footprints in each tile $S$:

$$w_i = \mathbb{E}[|A_n|]\ C_i\, S, \tag{6.25}$$

and its value $v_i$, i.e. the efficiency gained by caching the node (see Section 6.6), using $\mathcal{F}_n$ to refer to the footprint set at node $i$:

$$v_i = \max\left\{\text{cost}_\Sigma^{\text{direct}}(\mathcal{F}_n \mid \Theta_i) - \text{cost}_\Sigma^{\text{cache}}(\mathcal{F}_n \mid \Theta_i),\ 0\right\}. \tag{6.26}$$

As output of our optimization, we introduce the binary variable $x_i$ for every node $i$:

$$x_i = \begin{cases} 1 & \text{if node } i \text{ is cached,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.27}$$

**Constraints.** A solution to our optimization problem must not exceed the global available cache space $W$:

$$\sum_i w_i\, x_i \le W. \tag{6.28}$$

To implement our single-stage restriction, we compute pairwise conflicts of nodes. Nodes $i$ and $j$ are in conflict if there is any path from a source node to the output node that includes $i$ and $j$. We compute all conflicts using the reachability relation of all graphs, and add the following constraint for every conflict between a node $i$ and $j$:

$$x_i + x_j \le 1. \tag{6.29}$$

**Objective.** Under the above constraints, we seek to maximize evaluation efficiency by maximizing the total value

$$\sum_i v_i\, x_i. \tag{6.30}$$

## 6.9 Evaluation

**Implementation.** We provide source code of an implementation of our method [SHS*25a], which includes the procedural texture graph evaluation with caching and the prediction steps. To obtain footprints from a path tracer and for final renders, we use a CPU-based research rendering framework. We use HiGHS [HSF*25; HH18] to solve our ILP problem instances (see Section 6.8). It usually finds an optimal solution in the order of tens of milliseconds.

**Methodology.** We first record footprint requests when rendering each of the scenes, separated by material ID. We then run a single texture graph on footprints from one material ID at a time. By performing this for each available combination of scene, texture graph and material ID, we get a distribution of results. In some cases, we will report the distribution with one of these variables kept fixed. Since we are mainly interested in properties of footprint distribution and texture graphs and how

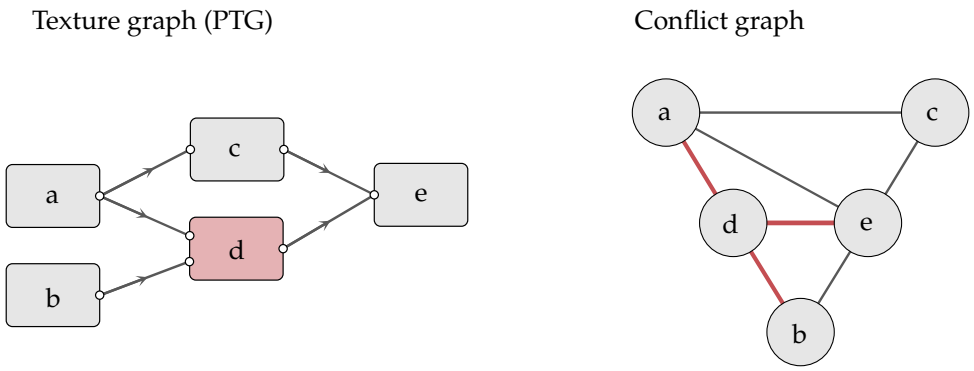



**Figure 6.8:** *Left: an example procedural texture graph with five nodes. Node d is marked as a cache node. Right: conflict graph, showing all nodes and edges between all nodes that are conflicted with respect to their cache state. Node b would be in conflict with d, since they share a sub-path on the way to the output node e, the edge between d and e in this example. Node c on the other hand is not in conflict with d and could be cached at the same time.*

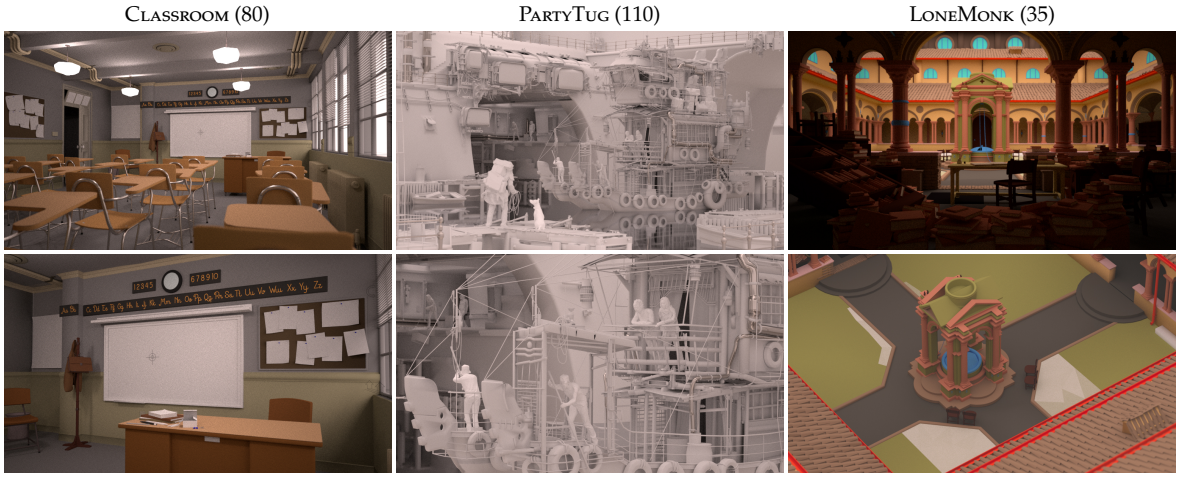



**Figure 6.9:** *Primary (top row) and secondary (bottom row) camera angle for the scenes used in our evaluation, with the number of materials next to scene names.*

they interact, we use our prediction results as a reference. We evaluate the accuracy of our introduced approximations separately in Figs. 6.5 to 6.7. We presume our cost measurement computation to be accurate enough to not skew results heavily if not using $\lambda_{\text{none}}$. When our tile number prediction fails, it mostly underestimates the total number of tiles, which would be an advantage for other caching methods as well, e.g. always caching the output node.

**Data.** We use the three scenes shown in Fig. 6.9 to evaluate our method. These are based on Blender demo files [Ble25]. We additionally use a secondary camera angle of each scene for some tests. We record footprints for an output resolution of $1280 \times 720$ and limit path length to 8 vertices. For texture graphs, we use the four procedural textures shown in Fig. 6.5, as well as two additional ones (TOOTH, CONCRETE). These were converted from example files from Adobe Substance Designer [Ado25] to our custom implementation and consist of 39–80 nodes.

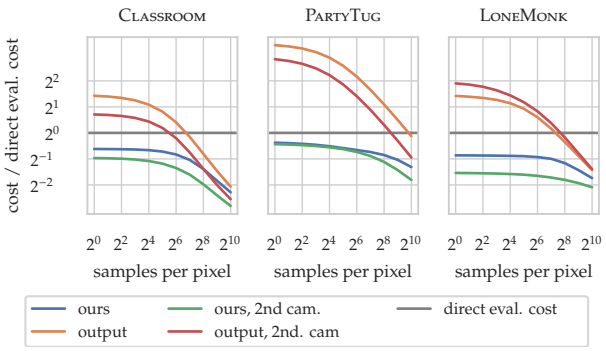



**Figure 6.10:** *Our method achieves lower total graph evaluation costs in the three scenes shown. Caching efficiency overall depends on camera angle, as the results with the secondary camera angle show.*

**Default parameters.** We set the parameters of our cost measure (see Section 6.6.1) to conform with our implementation. Since our cache reconstructs from isotropic footprints, we choose $\lambda_{\text{iso}} = 1$ for $\Theta_{\text{cache}}$. We also assume only isotropic filtering to be available for source nodes and set $\lambda_{\text{iso}} = 1$ for those, too. For inner nodes, we choose $\lambda_{\text{const}} = 1$, since they process footprints mostly independently of their shape or size in our graphs. We set the cost constant $c$ for each node based on benchmarking our implementation on footprint evaluation. We use a profiling phase of $k = 16$ samples per pixel (spp) for predicting tile numbers, and leave the cache capacity unconstrained ($W = \infty$) for optimization.

**Potential of caching.** In Fig. 6.11, we show the relative cost reduction of our approach compared to using direct evaluation of the texture graph. We compare this to the baseline method of caching requested texture tiles at the output node (*output*). While some texture graphs are more susceptible to caching than others, they display similar trends overall. Although caching will always pay off at some sample count, this can be quite high, as the *output* strategy shows. For a large share of material IDs, caching outputs will not amortize for the considered sample count of 1024, and even lead to an increase in evaluation cost by 2–4×. As the solution space of our optimization includes direct evaluation, it will never choose an option that leads to increased cost, given accurate predictions. It cannot improve results for very cacheable material IDs, since for these *output* is generally a good strategy. However, our median cost reduction is strictly better than caching outputs, although with varying significance depending on the graph.
Since relative cost reduction does not consider the significance of the material to the whole scene, we also show absolute cost reductions for whole scenes in Fig. 6.10. We average results for the available texture graphs. Here, our method can show a consistent cost improvement of 2× for lower sample count and up to 4× for higher sample count, while caching outputs only pays off at high sample counts, with still higher total cost. We presume the secondary cameras to perform better for caching output nodes in Classroom and PartyTug, since these are more close-up views of the scene.

**Limited cache capacity.** We investigate the effects of limiting cache capacity in Fig. 6.12. For each material ID and graph combination, we limit cache capacity in relation to the space required for caching requested texture tiles at the output node (*output*). Generally, our method is often able to find good strategies with cache space usage equal to *output*. It also benefits from a slightly increased cache

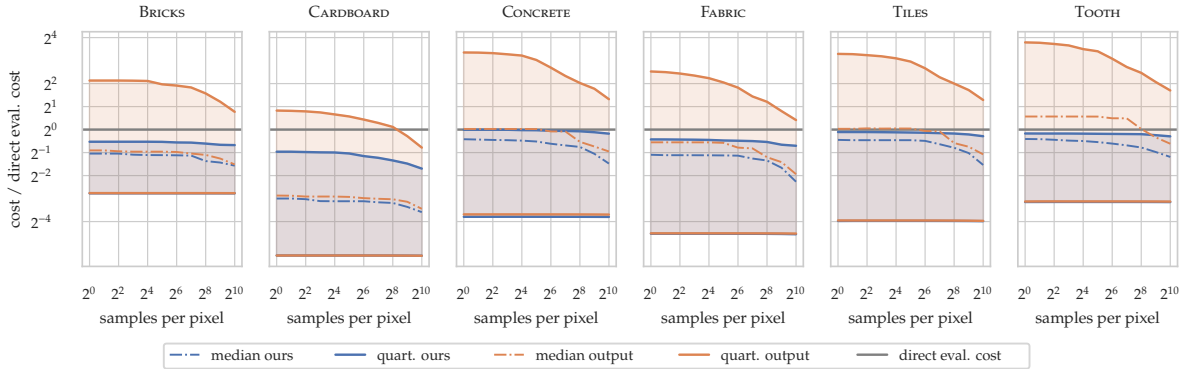



**Figure 6.11:** *Texture graphs vary in their suitability for caching. Generally, caching output nodes can achieve similar evaluation cost reduction as our method if the target sample count is sufficiently high, but can also result in increased cost for many material IDs. Our method avoids these cases and achieves lower cost on average.*

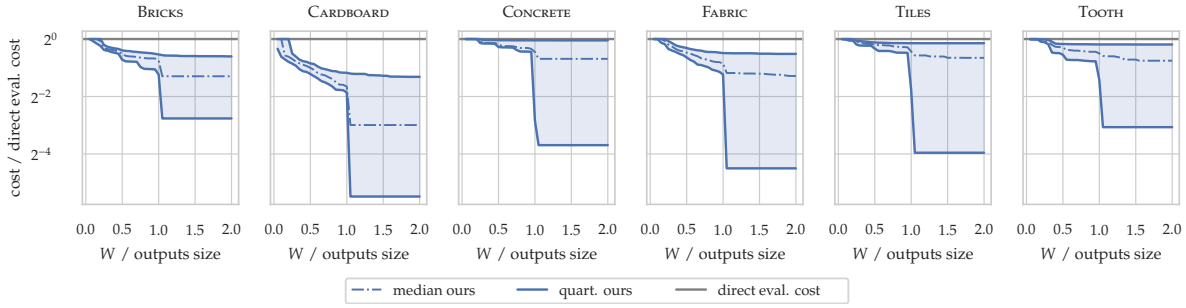



**Figure 6.12:** *Constraining the available cache space W reduces the benefit of caching, ultimately resulting in the same cost as direct evaluation of texture graphs. Our method is able to find compromises in between, trading an increase in cost for cache space. Providing more space than for caching output nodes does not seem to further increase efficiency.*

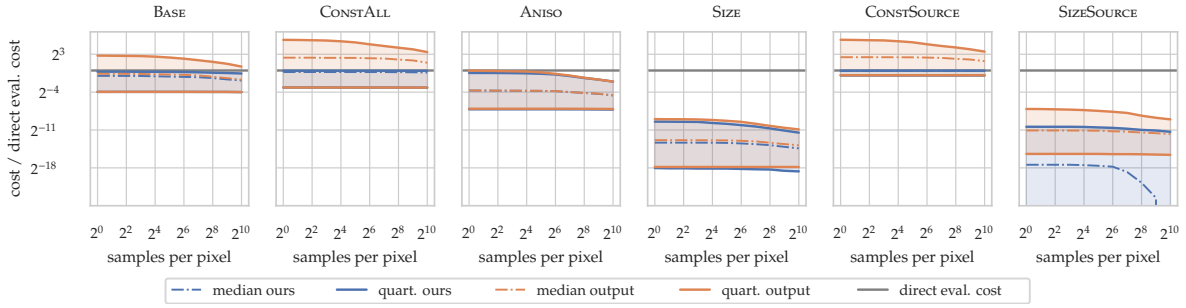



**Figure 6.13:** *Comparison of the effect of different parameters for our cost measure. Higher (ConstAll, ConstSource) or lower (Aniso, Size, SizeSource) relative cache fill cost make caching less or more attractive, respectively.*

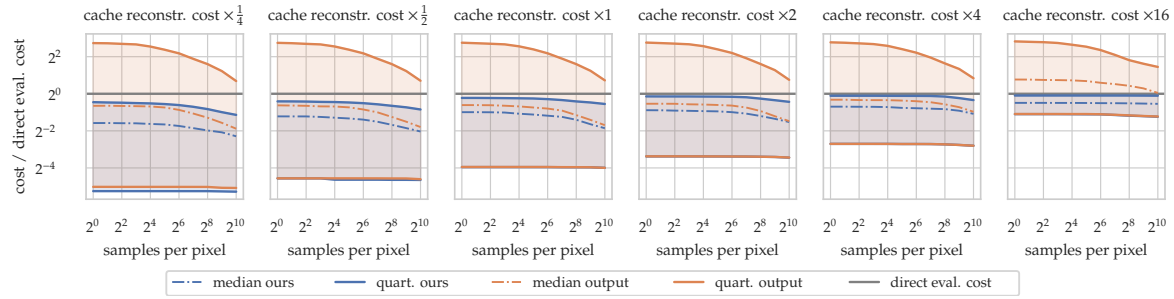



**Figure 6.14:** *Higher cost of reconstruction from cache reduces the best case benefits of caching overall. At the same time, worst case behavior of output caching is not affected much, since it is dominated by cache fill cost.*

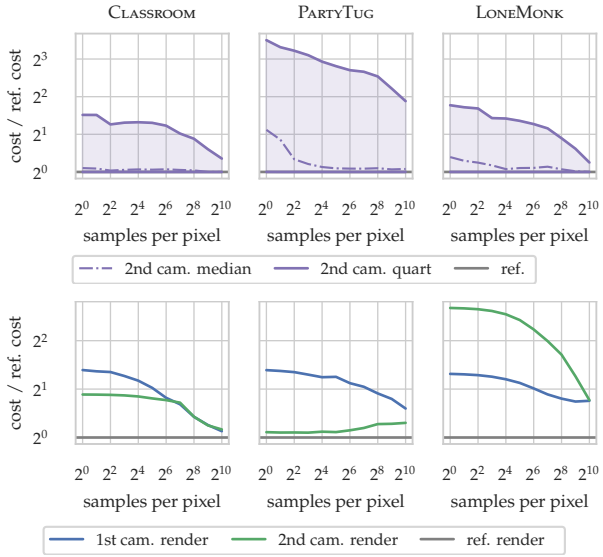



**Figure 6.15:** *Optimizing the evaluation strategy on a different footprint distribution leads to suboptimal results. Top: We optimize using our primary camera and render from the secondary camera. Relative cost of each material ID increases in many cases compared to optimizing on the secondary camera (ref.). Bottom: Total cost increases in the three scenes when using this setup. We also include the reverse case, i.e. optimizing on the secondary camera and rendering on the primary camera.*

capacity, although this trend does not seem to continue for further increases. For limited capacity, our method transitions smoothly to equal cost with direct evaluation, but is still able to improve upon it with little cache capacity.

**Cost measure parameters.** We compare the overall effect of different choices of cost coefficients (see Section 6.6.1) in Fig. 6.13. We describe the following options in terms of their difference to the default choice (Base), which is to set $\lambda_{\text{iso}} = 1$ for source nodes and cache reconstruction, $\lambda_{\text{const}} = 1$ for inner nodes:

ConstAll: $\lambda_{\text{const}} = 1$ for all nodes and cache reconstruction,

Aniso: $\lambda_{\text{iso}} = 1$ for inner nodes,

Size: $\lambda_{\text{none}} = 1$ for all nodes,

ConstSource: $\lambda_{\text{const}} = 1$ for source nodes,

SizeSource: $\lambda_{\text{none}} = 1$ for source nodes.

When comparing ConstAll to our default coefficients, there is only a slight shift with our method, while caching outputs is much worse overall. This could be explained by a higher cost to fill the cache compared to direct evaluation. Respectively, we observe that when inner nodes have a cost depending on footprint shape (Aniso) or size (Size), caching output nodes is generally a good and reliable strategy. In these cases, cache fill cost is low compared to direct evaluation. Varying cost at source nodes, ConstSource makes it more costly to cache footprints compared to direct evaluation, setting back amortization, while SizeSource has the opposite effect.
We also consider varying cache reconstruction cost in Fig. 6.14 by using a multiple of our default cache cost constant $c$. While reconstruction cost has a visible influence on the cost reduction that caching can achieve, it seems to not affect worst-case behavior. This could indicate that in these cases cost is dominated by the cache fill cost.

**Generalizability of solutions.** We investigate how specific our evaluation strategies are to a footprint distribution in Fig. 6.15. For this, we use our method to obtain an evaluation strategy on the primary camera and evaluate it on the footprints generated from the secondary camera. We compare the relative cost to an evaluation strategy that was optimized for the secondary camera. While many material IDs are not affected by the changed camera, in many cases cost is much higher without the specific optimization. We also show how the total cost of the scene is affected and include the reversed test of using the secondary camera to optimize. This demonstrates that the overall efficiency of evaluation strategies can depend significantly on the distribution of footprints.

## 6.10 Conclusion

We introduced a framework to examine the performance and resource usage of rendering systems with a procedural texture graph at the core of their shading system. We showed that the efficiency of a shading strategy and its evaluation crucially depends on the frequency, distribution, and shape

of the queries performed during rendering. These requests to the shading system originate from a path tracer and are decisively characterized by their pixel footprint. We provide a statistical prediction method to optimize the location of cache nodes in a procedural texture graph, taking performance, resource usage, and anticipated frequency of queries at a target sample rate into account. Our framework can indicate to rendering engineers whether shade-on-hit or precomputation are viable options for their application. Since our data collection and optimization scheme is relatively lightweight, it can also be implemented into an offline rendering system directly, to select optimal cache node locations. There are several directions for future research, including adaptation to GPU rendering and more advanced path sampling algorithms, as well as extension to the temporal domain. GPU architectures might demand changes to the cost model as well as to the implementation of our profiling phase to accommodate increased parallelism. More advanced sampling algorithms like path guiding could affect the accuracy of our prediction, since we assume path sampling to be invariable throughout rendering. The temporal domain might be interesting to support rendering movie sequences while exploiting shared, time-independent computations in the shading.

# 6.A Footprint derivation

## 6.A.1 Introduction

The footprint of a path for a vertex describes the distribution of its position, considering all random decisions along the way when assuming adjoint analog sampling. In practice, we will approximate this using a Gaussian distribution and use the footprint in texture space as input to our system.

In the following, we describe our approximate model for footprints (Section 6.A.2), which we base on the half vector space formulation of light transport [KHD14]. Using the slope space parameterization for half vectors allows us to easily incorporate a distribution of normals (NDF) [HKD15] based on a surface roughness estimate, which makes for a convenient and general BSDF approximation. Based on this model, we derive the equations for the footprint at a given path vertex (Section 6.A.3). The resulting recurrence can be computed incrementally in linear time, starting from a Gaussian distribution in screen space. We summarize our results in Algorithm 3, which shows how to use our derivations to compute footprints for a path.

## 6.A.2 Model

We base our derivation on a local first-order approximation of light transport around a base path. The footprint follows from statistical properties of nearby paths, when we interpret this model stochastically. Specifically, we analyze the distribution of paths that are sampled analogously to the approximation, starting from the camera. Such a random path depends only on the pairwise independently sampled screen space position and half vectors.

### 6.A.2.1 Preliminaries: half vector space

Given a base path of length $k+1$, we can parameterize it both as a sequence of vertices $x_0, \ldots, x_k$ or as a sequence of half vectors $h_1, \ldots, h_{k-1}$ and endpoints $x_0, x_k$ [KHD14]. As we want to relate changes in

**Algorithm 3** Footprint computation

**Inputs:** $\mathrm{var}(\Delta\mathbf{x}_1), \mathrm{var}(\Delta\mathbf{h}_1), \ldots, \mathrm{var}(\Delta\mathbf{h}_{k-1}), (A_i, B_i, C_i)_{1\le i\le k-1}$
**Outputs:** $\mathrm{var}(\Delta\mathbf{x}_2), \ldots, \mathrm{var}(\Delta\mathbf{x}_k)$

$\mathrm{var}(\Delta\mathbf{x}_2) \leftarrow C_1^{-1}\left(\mathrm{var}(\Delta\mathbf{h}_1) + B_1\,\mathrm{var}(\Delta\mathbf{x}_1)B_1^T\right)C_1^{-T}$
$\mathrm{cov}_{p,pp} \leftarrow -C_1^{-1}B_1\,\mathrm{var}\,(\Delta\mathbf{x}_1)$ ▷ *Eq.* (6.45)
$\mathrm{var}_{pp} \leftarrow \mathrm{var}(\Delta\mathbf{x}_1)$
$\mathrm{var}_p \leftarrow \mathrm{var}(\Delta\mathbf{x}_2)$
**for** $i \in \{2, \ldots, k-1\}$ **do**
  $\mathrm{cross\_cov} \leftarrow B_i\ \mathrm{cov}_{p,pp}\ A_i^T$
  $\mathrm{var}(\Delta\mathbf{g}_i) \leftarrow A_i\,\mathrm{var}_{pp}\,A_i^T + B_i\,\mathrm{var}_p\,B_i^T$
    $+\,\mathrm{cross\_cov} + \mathrm{cross\_cov}^T$ ▷ *Eq.* (6.44)
  $\mathrm{var}(\Delta\mathbf{x}_{i+1}) \leftarrow C_i^{-1}\left(\mathrm{var}(\Delta\mathbf{h}_i) + \mathrm{var}(\Delta\mathbf{g}_i)\right)C_i^{-T}$
  $\mathrm{cov}_{p,pp} \leftarrow -C_i^{-1}\left(A_i\,\mathrm{cov}_{p,pp}^T + B_i\,\mathrm{var}_p\right)$ ▷ *Eq.* (6.46)
  $\mathrm{var}_{pp} \leftarrow \mathrm{var}_p$
  $\mathrm{var}_p \leftarrow \mathrm{var}(\Delta\mathbf{x}_{i+1})$

half vector to changes in vertex position, we make use of the Jacobian matrix of the mapping from one parameterization to the other, which is known as the constraint derivative matrix [Jak13]. This has a block tridiagonal structure and can be defined by the submatrices

$$A_i = \frac{\partial h_i}{\partial x_{i-1}}, \quad B_i = \frac{\partial h_i}{\partial x_i}, \quad C_i = \frac{\partial h_i}{\partial x_{i+1}}. \tag{6.31}$$

We use the slope space parameterization for $\mathbf{h} \in \mathbb{R}^2$ and a local tangent frame for $\mathbf{x} \in \mathbb{R}^2$. As a first-order approximation, we use the constraint derivative matrix to map from changes in vertex position $\Delta\mathbf{x}$ to changes in half vector $\Delta\mathbf{h}$:

$$\begin{pmatrix} B_1 & C_1 & & & & 0 \\ \cdots & \cdots & \cdots & & & \\ & A_i & B_i & C_i & & \\ & & \cdots & \cdots & \cdots & \\ 0 & & & A_{k-1} & B_{k-1} & C_{k-1} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x}_1 \\ \cdots \\ \Delta\mathbf{x}_i \\ \cdots \\ \Delta\mathbf{x}_k \end{pmatrix} = \begin{pmatrix} \Delta\mathbf{h}_1 \\ \cdots \\ \Delta\mathbf{h}_i \\ \cdots \\ \Delta\mathbf{h}_{k-1} \end{pmatrix} \tag{6.32}$$

Please note that this matrix has one extra block column compared to the more common $(k-2)\times(k-2)$ block structure, since we assume only the beginning of the path to be fixed [Jak13, Sec. 4.3].

#### 6.A.2.2 Stochastic model

Given the random variables $\Delta\mathbf{x}_1$ and $\Delta\mathbf{h}_1, \ldots, \Delta\mathbf{h}_{k-1}$, we use Eq. (6.32) to compute all other $\Delta\mathbf{x}_2, \ldots, \Delta\mathbf{x}_k$, provided that all $C_1, \ldots, C_{k-1}$ are invertible. We proceed row-wise. The first row

$$B_1\,\Delta\mathbf{x}_1 + C_1\Delta\mathbf{x}_2 = \Delta\mathbf{h}_1 \tag{6.33}$$

yields a solution for $\Delta\mathbf{x}_2$:

$$\Delta\mathbf{x}_2 = C_1^{-1}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1\right). \tag{6.34}$$

Similarly, we obtain a solution for $\Delta\mathbf{x}_3, \ldots, \Delta\mathbf{x}_{k-1}$ using

$$\Delta\mathbf{x}_{i+1} = C_i^{-1} \left(\Delta\mathbf{h}_i - A_i \Delta\mathbf{x}_{i-1} - B_i \Delta\mathbf{x}_i\right). \tag{6.35}$$

## 6.A.3 Derivation

Since we approximate the footprint as a Gaussian distribution, it is sufficient to derive a covariance matrix of each distribution of vertex position. In the following, we will use our model to derive these covariances based on known $\mathrm{var}(\Delta\mathbf{x}_1)$ and $\mathrm{var}(\Delta\mathbf{h}_1), \ldots, \mathrm{var}(\Delta\mathbf{h}_{k-1})$. We derive $\mathrm{var}(\Delta\mathbf{x}_1)$ by projecting the pixel filter, for which we use a standard deviation of half a pixel, to the first vertex using ray differentials. For $\mathrm{var}(\Delta\mathbf{h}_i)$, we simply convert the Beckmann-equivalent roughness $\alpha$ to a standard deviation in slope space $\sigma = \frac{\alpha}{\sqrt{2}}$.

### 6.A.3.1 Preliminaries: properties of cross-covariance

We make use of the following basic properties of cross-covariance:

$$\mathrm{var}(\mathbf{X}) = \mathrm{cov}(\mathbf{X}, \mathbf{X}), \tag{6.36}$$

$$\mathrm{var}(\mathbf{X} + \mathbf{Y}) = \mathrm{var}(\mathbf{X}) + \mathrm{var}(\mathbf{Y}) + \mathrm{cov}(\mathbf{X}, \mathbf{Y}) + \mathrm{cov}(\mathbf{Y}, \mathbf{X}), \tag{6.37}$$

$$\mathrm{cov}(\mathbf{X}, \mathbf{Y}) = \mathrm{cov}(\mathbf{Y}, \mathbf{X})^T, \tag{6.38}$$

$$\mathrm{cov}(\mathbf{X} + \mathbf{Y}, \mathbf{Z}) = \mathrm{cov}(\mathbf{X}, \mathbf{Z}) + \mathrm{cov}(\mathbf{Y}, \mathbf{Z}), \tag{6.39}$$

$$\mathrm{cov}(A\mathbf{X}, B\mathbf{Y}) = A\,\mathrm{cov}(\mathbf{X}, \mathbf{Y})B^T, \tag{6.40}$$

where $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are random variables and $A, B$ are matrices.

### 6.A.3.2 Propagating covariance

Using these properties, we derive covariances of any $\mathrm{var}(\Delta\mathbf{x}_i)$ with $2 \leq i \leq k$. Please note that the following equations hold even without assuming Gaussianity. We first prove the following lemma, to simplify our derivation later on.

**Lemma.** Any change in half vector $\Delta\mathbf{h}_j$ is uncorrelated with any change in vertex position that does not occur after it on the path:

$$\forall i \in \{1, \ldots, j\}\colon\ \mathrm{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{h}_j) = 0. \tag{6.41}$$

*Proof.* By induction. The base case $i = 1$ follows by definition, since $\Delta\mathbf{x}_1$ and $\Delta\mathbf{h}_1, \ldots, \Delta\mathbf{h}_{k-1}$ are pairwise independent:

$$\mathrm{cov}(\Delta\mathbf{x}_1, \Delta\mathbf{h}_j) = 0.$$

For the inductive step ($i \to i + 1$), we plug in Eqs. (6.34) and (6.35).

*Case* $i + 1 = 2$.

$$\begin{aligned}\operatorname{cov}(\Delta\mathbf{x}_2, \Delta\mathbf{h}_j) &= \operatorname{cov}(C_1^{-1}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1\right), \Delta\mathbf{h}_j) \\ &= C_1^{-1}\operatorname{cov}(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1, \Delta\mathbf{h}_j) \\ &= C_1^{-1}(\underbrace{\operatorname{cov}(\Delta\mathbf{h}_1, \Delta\mathbf{h}_j)}_{\substack{=0 \\ \text{by definition}}} - B_1 \underbrace{\operatorname{cov}(\Delta\mathbf{x}_1, \Delta\mathbf{h}_j)}_{\substack{=0 \\ \text{induct. hypothesis}}}) = 0\end{aligned}$$

*Case* $i + 1 > 2$.

$$\begin{aligned}&\operatorname{cov}(\Delta\mathbf{x}_{i+1}, \Delta\mathbf{h}_j) = \operatorname{cov}(C_i^{-1}\left(\Delta\mathbf{h}_i - A_i\Delta\mathbf{x}_{i-1} - B_i\Delta\mathbf{x}_i\right), \Delta\mathbf{h}_j) \\ &= C_i^{-1}\operatorname{cov}(\Delta\mathbf{h}_i - A_i\Delta\mathbf{x}_{i-1} - B_i\Delta\mathbf{x}_i, \Delta\mathbf{h}_j) \\ &= C_i^{-1}(\underbrace{\operatorname{cov}(\Delta\mathbf{h}_i, \Delta\mathbf{h}_j)}_{\substack{=0 \\ \text{by definition}}} - A_i \underbrace{\operatorname{cov}(\Delta\mathbf{x}_{i-1}, \Delta\mathbf{h}_j)}_{\substack{=0 \\ \text{induct. hypothesis}}} - B_i \underbrace{\operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{h}_j)}_{\substack{=0 \\ \text{induct. hypothesis}}}) \\ &= 0\end{aligned}$$

□

**Footprint for $x_2$.**

$$\begin{aligned}\operatorname{var}(\Delta\mathbf{x}_2) &= \operatorname{var}\left(C_1^{-1}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1\right)\right) && \text{Eq. (6.34)} \\ &= C_1^{-1}\operatorname{var}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1\right)C_1^{-T} \\ &= C_1^{-1}\left(\operatorname{var}(\Delta\mathbf{h}_1) + \operatorname{var}(-B_1\Delta\mathbf{x}_1)\right)C_1^{-T} && \text{Eq. (6.41)} \\ &= C_1^{-1}\left(\operatorname{var}(\Delta\mathbf{h}_1) + B_1\operatorname{var}(\Delta\mathbf{x}_1)B_1^T\right)C_1^{-T}\end{aligned} \tag{6.42}$$

**Footprint for $x_{i+1},\ i > 1$.** Since $\operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{x}_{i-1}) \neq 0$, the case for any following vertex $x_{i+1}$ is slightly more involved:

$$\begin{aligned}\operatorname{var}(\Delta\mathbf{x}_{i+1}) &= \operatorname{var}\left(C_i^{-1}\left(\Delta\mathbf{h}_i - A_i\Delta\mathbf{x}_{i-1} - B_i\Delta\mathbf{x}_i\right)\right) && \text{Eq. (6.35)} \\ &= C_i^{-1}\left(\operatorname{var}(\Delta\mathbf{h}_i) + \operatorname{var}(-\Delta\mathbf{g}_i)\right)C_i^{-T} && \text{Eq. (6.41)} \\ &= C_i^{-1}\left(\operatorname{var}(\Delta\mathbf{h}_i) + \operatorname{var}(\Delta\mathbf{g}_i)\right)C_i^{-T},\end{aligned} \tag{6.43}$$

where we defined $\Delta\mathbf{g}_i \coloneqq A_i\Delta\mathbf{x}_{i-1} + B_i\Delta\mathbf{x}_i$.
To compute $\operatorname{var}(\Delta\mathbf{g}_i)$, we expand terms:

$$\begin{aligned}&\operatorname{var}(\Delta\mathbf{g}_i) = \operatorname{var}\left(A_i\Delta\mathbf{x}_{i-1} + B_i\Delta\mathbf{x}_i\right) \\ &= A_i\operatorname{var}(\Delta\mathbf{x}_{i-1})A_i^T + B_i\operatorname{var}(\Delta\mathbf{x}_i)B_i^T \\ &\quad + A_i\operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{x}_{i-1})^T B_i^T + B_i\operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{x}_{i-1})A_i^T\end{aligned} \tag{6.44}$$

of which the remaining unknown term is the cross-covariance $\operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{x}_{i-1})$. We again treat the cases $i = 2$ and $i > 2$ separately and expand using Eqs. (6.34) and (6.35).

$$\begin{aligned}\operatorname{cov}(\Delta\mathbf{x}_2, \Delta\mathbf{x}_1) &= \operatorname{cov}\left(C_1^{-1}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1\right), \Delta\mathbf{x}_1\right) \\ &= C_1^{-1}\operatorname{cov}\left(\Delta\mathbf{h}_1 - B_1\Delta\mathbf{x}_1, \Delta\mathbf{x}_1\right) \\ &= C_1^{-1}\left(\operatorname{cov}\left(\Delta\mathbf{h}_1, \Delta\mathbf{x}_1\right) + \operatorname{cov}\left(-B_1\Delta\mathbf{x}_1, \Delta\mathbf{x}_1\right)\right) \\ &= C_1^{-1}\operatorname{cov}\left(-B_1\Delta\mathbf{x}_1, \Delta\mathbf{x}_1\right) = -C_1^{-1}B_1\operatorname{var}\left(\Delta\mathbf{x}_1\right).\end{aligned} \tag{6.45}$$

$$
\begin{aligned}
&\forall i > 2\colon\ \operatorname{cov}(\Delta\mathbf{x}_i, \Delta\mathbf{x}_{i-1}) \\
&= \operatorname{cov}\left(C_{i-1}^{-1}\left(\Delta\mathbf{h}_{i-1} - A_{i-1}\Delta\mathbf{x}_{i-2} - B_{i-1}\Delta\mathbf{x}_{i-1}\right), \Delta\mathbf{x}_{i-1}\right) \\
&= C_{i-1}^{-1}\left(\operatorname{cov}\left(-A_{i-1}\Delta\mathbf{x}_{i-2}, \Delta\mathbf{x}_{i-1}\right) + \operatorname{cov}\left(-B_{i-1}\Delta\mathbf{x}_{i-1}, \Delta\mathbf{x}_{i-1}\right)\right) \\
&= C_{i-1}^{-1}\left(-A_{i-1}\operatorname{cov}\left(\Delta\mathbf{x}_{i-2}, \Delta\mathbf{x}_{i-1}\right) - B_{i-1}\operatorname{var}\left(\Delta\mathbf{x}_{i-1}\right)\right) \\
&= -C_{i-1}^{-1}\left(A_{i-1}\operatorname{cov}\left(\Delta\mathbf{x}_{i-1}, \Delta\mathbf{x}_{i-2}\right)^T + B_{i-1}\operatorname{var}\left(\Delta\mathbf{x}_{i-1}\right)\right).
\end{aligned}
\tag{6.46}
$$

We apply this equation recursively to arrive at a solution. We show how to iteratively compute footprints for a path in Algorithm 3.

# 7 Conclusion

In this dissertation, we investigated several approaches toward improving the efficiency and robustness of Monte Carlo-based light transport simulation for physically based rendering. While Monte Carlo integration provides a flexible framework for light transport simulation, its practical applicability is often limited by high variance. The robustness of sampling techniques has to be balanced with their computational cost, making a scene-independent choice challenging. Complex light transport effects inherently require global information for their precise sampling. In addition, algorithmic choices affecting the computational cost of path evaluations also depend on the rendered scene. Data-driven methods address these challenges and help to achieve convergence within acceptable computation time. The main contributions of this work focus on guiding and optimizing sampling and evaluation based on information gathered during rendering.

We introduced a general path guiding model that represents dependencies between successive interactions using joint distributions of path vertex triplets. This model supports both surface and volumetric transport and enables bidirectional guided path construction. We demonstrated its utility for learning and sampling challenging light transport, such as directional dependencies between glossy surface scattering and volumetric transport including distance sampling. Though our model was designed to be as simple as possible while remaining general, scenes with simpler configurations can still benefit from the reduced generality of even simpler models. Future improvements of our approach are likely possible, for example by explicit selection of model complexity in the learning phase, or by making use of more informative priors.

We also presented a more classical sampling technique to connect endpoints of subpaths in dense, forward-scattering participating media via a *bridge*. The defining property of a bridge is that it introduces an arbitrary number of intermediate vertices, while importance sampling all reciprocal squared distance and internal phase function terms. We showed the clear benefit of this technique in homogeneous volumes for scattering orders of up to 100 extra vertices. The technique is specialized in that it considers only part of the true measurement contribution. For example, heterogeneous media or emission distribution functions can break its assumptions. However, due to its simultaneous simplicity and effectiveness, it is still compelling for augmenting more general sampling approaches.

Further, we contributed to two theoretical frameworks that facilitate the use of Monte Carlo-based rendering in interactive applications on GPU hardware: Markov chain mixture models and ReSTIR. We believe that further exploration of similar ideas, especially with consideration of the parallelism of GPU hardware, can be fruitful for the realization of the full potential of interactive Monte Carlo rendering.

Aside from variance reduction, we also addressed the computational cost of path evaluation using a data-driven approach. We proposed a cost-based framework for optimizing the selection of nodes to cache in procedural texture graphs. We introduced a statistical model to estimate scene-specific costs and optimize the cache node selection using integer linear programming. We showed consistent

reductions in cost for several combinations of texture graphs and scenes, avoiding severe worst-case behavior in particular. To the best of our knowledge, our work provides the first principled treatment of this problem and opens up several directions for future research. While extensions, e.g. to the temporal domain, are appealing, an even more impactful research goal could be its simplification to allow for easier implementation.

In conclusion, this dissertation demonstrates that incorporating data-driven guidance can improve the robustness and efficiency of Monte Carlo-based rendering algorithms by reducing both variance and computational cost. The developed methods enable the simulation of more complex light transport phenomena and scene models than was previously feasible. Conceiving a unified rendering algorithm that robustly combines specialized sampling techniques with data-driven approaches, without incurring significant overhead, remains a difficult challenge for future work. We anticipate that the insights gained in this work can inform future approaches working toward this goal.

# Bibliography


[ABW14] Ament, Marco, Bergmann, Christoph, and Weiskopf, Daniel. "Refractive Radiative Transfer Equation". *ACM Transactions on Graphics* 33.2 (Apr. 2014), 17:1–17:22. doi: 10/gbf323 13.

[ÁBWM16] Áfra, Attila, Benthin, Carsten, Wald, Ingo, and Munkberg, Jacob. "Local Shading Coherence Extraction for SIMD-efficient Path Tracing on CPUs". *Proceedings of High Performance Graphics*. 2016. doi: 10.2312/hpg.20161198 89.

[Ado25] Adobe. *Substance 3D*. 2025. url: https://www.adobe.com/products/substance3d.html 87, 105.

[AHD25] Alber, Lucas, Hanika, Johannes, and Dachsbacher, Carsten. "Real-Time Markov Chain Path Guiding for Global Illumination and Single Scattering". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 8.1 (2025). doi: 10.1145/3728296 78.

[AK90] Arvo, James Richard and Kirk, David. "Particle Transport and Image Synthesis". *Computer Graphics (Proceedings of SIGGRAPH)* 24.4 (Sept. 1, 1990), 63–66. doi: 10/dtp6gd 20.

[Aka73] Akaike, H. "Information Theory and an Extension of the Maximum Likelihood Principle". *Second International Symposium on Information Theory.* Akademia Kiado. 1973, 267–281 42.

[Ama84] Amanatides, John. "Ray Tracing with Cones". *Computer Graphics (Proceedings of SIGGRAPH)* 18.3 (July 1, 1984), 129–135. doi: 10/b64smh 89.

[ANA*19] Akenine-Möller, Tomas, Nilsson, Jim, Andersson, Magnus, Barré-Brisebois, Colin, Toth, Robert, and Karras, Tero. "Texture Level of Detail Strategies for Real-Time Ray Tracing". *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. 2019, 321–345. doi: 10.1007/978-1-4842-4427-2_20 89.

[Arv01] Arvo, James Richard. "Stratified Sampling of 2-Manifolds". *State of the Art in Monte Carlo Global Illumination*. Vol. 29. ACM SIGGRAPH Courses. Aug. 2001. Chap. 4 22.

[Arv86] Arvo, James. "Backward Ray Tracing". *Developments in Ray Tracing*. Vol. 12. ACM SIGGRAPH Courses. Aug. 1986, 259–263 22.

[Arv95a] Arvo, James R. "Stratified Sampling of Spherical Triangles". *Annual Conference Series (Proceedings of SIGGRAPH)*. Aug. 1995, 437–438. doi: 10/bdrqbf 22.

[Arv95b] Arvo, James Richard. "Analytic Methods for Simulated Light Transport". PhD thesis. Yale University, Dec. 1995 10.

[BAC*18] Burley, Brent, Adler, David, Chiang, Matt Jen-Yuan, Driskill, Hank, Habel, Ralf, Kelly, Patrick, Kutz, Peter, Li, Yining Karl, and Teece, Daniel. "The Design and Evolution of Disney's Hyperion Renderer". *ACM Transactions on Graphics* 37.3 (July 2018), 33:1–33:22. doi: 10/gfjm8w 81, 88.

[Bil95] Billingsley, Patrick. *Probability and Measure*. John Wiley & Sons, 1995. isbn: 978-0-471-00710-4 7.

[BJNJ17] Bitterli, Benedikt, Jakob, Wenzel, Novák, Jan, and Jarosz, Wojciech. "Reversible Jump Metropolis Light Transport Using Inverse Mappings". *ACM Transactions on Graphics* 37.1 (2017), 1:1–1:12. doi: 10/gd52ph 25, 82, 85.

[Ble25] Blender Foundation. *Blender*. 2025. url: https://www.blender.org 87, 105.

[BMDS19] Bako, Steve, Meyer, Mark, DeRose, Tony, and Sen, Pradeep. "Offline Deep Importance Sampling for Monte Carlo Path Tracing". *Computer Graphics Forum (Proceedings of Pacific Graphics 2019)* 38.7 (2019), 527–542 29.

[BN12] Bruneton, Eric and Neyret, Fabrice. "A Survey of Non-Linear Pre-Filtering Methods for Efficient and Accurate Surface Shading". *IEEE Transactions on Visualization and Computer Graphics* 18.2 (Feb. 2012), 242–260. doi: 10/chq5qh 89.

[BN76] Blinn, James F. and Newell, Martin E. "Texture and Reflection in Computer Generated Images". *Communications of the ACM* 19.10 (Oct. 1976), 542–547. doi: 10/dct2v6 12.

[BRM*18] Bitterli, Benedikt, Ravichandran, Srinath, Müller, Thomas, Wrenninge, Magnus, Novák, Jan, Marschner, Steve, and Jarosz, Wojciech. "A Radiative Transfer Framework for Non-Exponential Media". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37.6 (Nov. 2018), 225:1–225:17. DOI: 10/gfz2cm 13.

[BSH02] Bekaert, Philippe, Sbert, Mateu, and Halton, John. "Accelerating Path Tracing by Re-Using Paths". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. June 2002. DOI: 10/ggdwkn 25.

[BWP*20] Bitterli, Benedikt, Wyman, Chris, Pharr, Matt, Shirley, Peter, Lefohn, Aaron, and Jarosz, Wojciech. "Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: 10/gg8xc7 25, 82.

[BYRN17] Belcour, Laurent, Yan, Ling-Qi, Ramamoorthi, Ravi, and Nowrouzezahrai, Derek. "Antialiasing Complex Global Illumination Effects in Path-Space". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (Jan. 2017). DOI: 10/ggfg5m 89.

[CCC87] Cook, Robert L., Carpenter, Loren, and Catmull, Edwin. "The Reyes Image Rendering Architecture". *Computer Graphics (Proceedings of SIGGRAPH)* 21.4 (1987), 95–102. DOI: 10/bk6dpr 88.

[CCY67] Coveyou, R. R., Cain, V. R., and Yost, K. J. "Adjoint and Importance in Monte Carlo Application". *Nuclear Science and Engineering* 27.2 (1967), 219–234. DOI: 10/gfzq75 20.

[CDK*09] Cohen, Edith, Duffield, Nick, Kaplan, Haim, Lund, Carsten, and Thorup, Mikkel. "Stream Sampling for Variance-Optimal Estimation of Subset Sums". *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '09. New York, 2009, 1255–1264 38.

[CFS*18] Christensen, Per, Fong, Julian, Shade, Jonathan, Wooten, Wayne, Schubert, Brenden, Kensler, Andrew, Friedman, Stephen, Kilpatrick, Charlie, Ramshaw, Cliff, Bannister, Marc, Rayner, Brenton, Brouillat, Jonathan, and Liani, Max. "RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering". *ACM Transactions on Graphics* 37.3 (Aug. 2018), 30:1–30:21. DOI: 10/gfznbs 81, 88, 89.

[DBLW15] Dorn, Jonathan, Barnes, Connelly, Lawrence, Jason, and Weimer, Westley. "Towards Automatic Band-Limited Procedural Shaders". *Computer Graphics Forum* 34.7 (2015), 77–87. DOI: 10.1111/cgf.12747 89.

[dEo16] d'Eon, Eugene. *A Hitchhiker's Guide to Multiple Scattering*. 2016. URL: http://www.eugenedeon.com/hitchhikers 54.

[dEon18] D'Eon, Eugene. "A Reciprocal Formulation of Nonexponential Radiative Transfer. 1: Sketch and Motivation". *Journal of Computational and Theoretical Transport* 47.1–3 (Apr. 16, 2018), 84–115. DOI: 10/gg5kzj 13.

[Dev86] Devroye, Luc. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986. DOI: 10.1007/978-1-4613-8643-8 71.

[DF88] Dyer, Martin E. and Frieze, Alan M. "On the complexity of computing the volume of a polyhedron". *SIAM Journal on Computing* 17.5 (1988), 967–974 60.

[DGJ*20] Diolatzis, Stavros, Gruson, Adrien, Jakob, Wenzel, Nowrouzezahrai, Derek, and Drettakis, George. "Practical Product Path Guiding Using Linearly Transformed Cosines". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* (2020). DOI: 10.1111/cgf.14051 29.

[DHB17] Dupuy, Jonathan, Heitz, Eric, and Belcour, Laurent. "A spherical cap preserving parameterization for spherical distributions". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (July 2017). DOI: 10.1145/3072959.3073694 22.

[DHI*13] Dupuy, Jonathan, Heitz, Eric, Iehl, Jean-Claude, Poulin, Pierre, Neyret, Fabrice, and Ostromoukhov, Victor. "Linear Efficient Antialiased Displacement and Reflectance Mapping". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 32.6 (Nov. 2013), 211:1–211:11. DOI: 10/gbd53f 92.

[DLR77] Dempster, A. P., Laird, N. M., and Rubin, D. B. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (Sept. 1, 1977), 1–22. DOI: 10/gfxzrv 40, 101.

[DLW93] Dutré, Philip, Lafortune, Eric P., and Willems, Yves D. "Monte Carlo light tracing with direct computation of pixel intensities". *Proceedings of the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*. Dec. 1993, 128–137 22.

[DPÖM21] Dodik, Ana, Papas, Marios, Öztireli, Cengiz, and Müller, Thomas. "Path Guiding Using Spatio-Directional Mixture Models". *Computer Graphics Forum* 41.1 (2021), 172–189. DOI: 10.1111/cgf.14428 29, 42, 43.

[DSH*23] Dittebrandt, Addis, Schüßler, Vincent, Hanika, Johannes, Herholz, Sebastian, and Dachsbacher, Carsten. "Markov Chain Mixture Models for Real-Time Direct Illumination". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 42.4 (2023). doi: 10.1111/cgf.14881 4, 77, 78.

[DWWH20] Deng, Hong, Wang, Beibei, Wang, Rui, and Holzschuch, Nicolas. "A Practical Path Guiding Method for Participating Media". *Computational Visual Media* 6 (Mar. 2020), 37–51. doi: 10.1007/s41095-020-0160-1 30.

[EK18] Estevez, Alejandro Conty and Kulla, Christopher. "Importance Sampling of Many Lights with Adaptive Tree Splitting". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.2 (Aug. 2018), 25:1–25:17. doi: 10/ggh89v 22.

[ENSB13] Eisenacher, Christian, Nichols, Gregory, Selle, Andrew, and Burley, Brent. "Sorted Deferred Shading for Production Path Tracing". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 32.4 (2013), 125–132. doi: 10/gfzv8r 89.

[FH22] Fujieda, Shin and Harada, Takahiro. "Progressive Material Caching". *SIGGRAPH Asia 2022 Technical Communications*. Nov. 22, 2022. doi: 10.1145/3550340.3564223 90.

[FHL*18] Fascione, Luca, Hanika, Johannes, Leone, Mark, Droske, Marc, Schwarzhaupt, Jorge, Davidovič, Tomáš, Weidlich, Andrea, and Meng, Johannes. "Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production". *ACM Transactions on Graphics* 37.3 (Aug. 2018), 31:1–31:18. doi: 10/gfznbt 88, 89.

[FS24] Fournier, Romain and Sauvage, Basile. "Mix-Max: A Content-Aware Operator for Real-Time Texture Transitions". *Computer Graphics Forum* 43.6 (Sept. 2024). doi: 10.1111/cgf.15193 89.

[GBBE18] Guo, Jerry, Bauszat, Pablo, Bikker, Jacco, and Eisemann, Elmar. "Primary Sample Space Path Guiding". *Eurographics Symposium on Rendering - EI & I*. July 2018, 73–82. doi: 10.2312/sre.20181174 29.

[GH86] Greene, Ned and Heckbert, Paul S. "Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter". *IEEE Computer Graphics and Applications* 6.6 (1986), 21–27. doi: 10.1109/MCG.1986.276738 89, 93.

[GHS*22] Guerrero, Paul, Hašan, Miloš, Sunkavalli, Kalyan, Měch, Radomír, Boubekeur, Tamy, and Mitra, Niloy J. "MatFormer: A Generative Model for Procedural Materials". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 22, 2022), 46:1–46:12. doi: 10/gqjn68 90.

[GIF*18] Georgiev, Iliyan, Ize, Thiago, Farnsworth, Mike, Montoya-Vozmediano, Ramón, King, Alan, Lommel, Brecht Van, Jimenez, Angel, Anson, Oscar, Ogaki, Shinji, Johnston, Eric, Herubel, Adrien, Russell, Declan, Servant, Frédéric, and Fajardo, Marcos. "Arnold: A Brute-Force Production Path Tracer". *ACM Transactions on Graphics* 37.3 (Aug. 2018), 32:1–32:12. doi: 10/gfznb2 88.

[GKDS12] Georgiev, Iliyan, Křivánek, Jaroslav, Davidovič, Tomáš, and Slusallek, Philipp. "Light Transport Simulation with Vertex Connection and Merging". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31.6 (Nov. 2012), 192:1–192:10. doi: 10/gbb6q7 24.

[GKH*13] Georgiev, Iliyan, Křivánek, Jaroslav, Hachisuka, Toshiya, Nowrouzezahrai, Derek, and Jarosz, Wojciech. "Joint Importance Sampling of Low-Order Volumetric Scattering". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 32.6 (Nov. 2013), 1–14. doi: 10/gbd5qs 53.

[Gla95] Glassner, Andrew S. "A Model for Fluorescence and Phosphorescence". *Photorealistic Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. 1995, 60–70. doi: 10.1007/978-3-642-87825-1_5 13.

[GMH*19] Georgiev, Iliyan, Misso, Zackary, Hachisuka, Toshiya, Nowrouzezahrai, Derek, Křivánek, Jaroslav, and Jarosz, Wojciech. "Integral Formulations of Volumetric Transmittance". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Nov. 8, 2019), 154:1–154:17. doi: 10/dffn 60.

[Gri08] Gritz, Larry. *OpenImageIO*. 2008. url: https://github.com/AcademySoftwareFoundation/OpenImageIO 89.

[GSDT22] Grenier, C., Sauvage, B., Dischler, J-M., and Thery, S. "Color-mapped noise vector fields for generating procedural micro-patterns". *Computer Graphics Forum (Proceedings of Pacific Graphics)* 41.7 (2022), 477–487 89, 92.

[GSKC10] Gritz, Larry, Stein, Clifford, Kulla, Chris, and Conty, Alejandro. "Open Shading Language". *ACM SIGGRAPH Talks*. 2010, 1–1. doi: 10.1145/1837026.1837070 88–90.

[GYGS22] GRITTMANN, PASCAL, YAZICI, ÖMERCAN, GEORGIEV, ILIYAN, and SLUSALLEK, PHILIPP. "Efficiency-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 2022). DOI: 10.1145/3528223.3530126 24.

[Has70] HASTINGS, WILFRED K. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika* 57.1 (Apr. 1, 1970), 97–109. DOI: 10/dkbmcf 25.

[HD14] HANIKA, JOHANNES and DACHSBACHER, CARSTEN. "Efficient Monte Carlo Rendering with Realistic Lenses". *Computer Graphics Forum (Proceedings of Eurographics)* 33.2 (2014), 323–332. DOI: 10/f57czp 13.

[HDF15] HANIKA, JOHANNES, DROSKE, MARC, and FASCIONE, LUCA. "Manifold Next Event Estimation". *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34.4 (June 2015), 87–97. DOI: 10.1111/cgf.12681 23, 54.

[HDM76] HOROWITZ, Y. S., DUBI, A., and MORDECHAI, S. "A New Direct Point Flux Monte Carlo Estimator Using Compensated Rotational Symmetry". *Nuclear Science and Engineering* 59.4 (1976), 427–429. DOI: 10.13182/NSE76-A26842 71, 72.

[Hec89] HECKBERT, PAUL. *Fundamentals of Texture Mapping and Image Warping*. Tech. rep. UCB/CSD-89-516. June 1989. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1989/5504.html 89.

[Hec90] HECKBERT, PAUL S. "Adaptive Radiosity Textures for Bidirectional Ray Tracing". *Computer Graphics (Proceedings of SIGGRAPH)* 24.4 (Aug. 1990), 145–154. DOI: 10/bsxgp4 22.

[HEV*16] HERHOLZ, SEBASTIAN, ELEK, OSKAR, VORBA, JIŘÍ, LENSCH, HENDRIK, and KŘIVÁNEK, JAROSLAV. "Product Importance Sampling for Light Transport Path Guiding". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 35.4 (2016), 25:1–25:19. DOI: 10/f842dt 29, 54.

[HGP*19] HUA, BINH-SON, GRUSON, ADRIEN, PETITJEAN, VICTOR, ZWICKER, MATTHIAS, NOWROUZEZAHRAI, DEREK, EISEMANN, ELMAR, and HACHISUKA, TOSHIYA. "A Survey on Gradient-Domain Rendering". *Computer Graphics Forum (Proceedings of Eurographics State of the Art Reports)* 38.2 (2019), 455–472. DOI: 10/ggd8m5 25, 82.

[HH18] HUANGFU, Q. and HALL, J. A. J. "Parallelizing the dual revised simplex method". *Mathematical Programming Computation* 10 (2018), 119–142. DOI: 10.1007/s12532-017-0130-5 104.

[HKD14] HACHISUKA, TOSHIYA, KAPLANYAN, ANTON S., and DACHSBACHER, CARSTEN. "Multiplexed Metropolis Light Transport". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014), 100:1–100:10. DOI: 10/f6cswv 25.

[HKD15] HANIKA, JOHANNES, KAPLANYAN, ANTON, and DACHSBACHER, CARSTEN. "Improved Half Vector Space Light Transport". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 34.4 (July 1, 2015), 65–74. DOI: 10/gfzv83 25, 89, 110.

[HL90] HANRAHAN, PAT and LAWSON, JIM. "A Language for Shading and Lighting Calculations". *Computer Graphics (Proceedings of SIGGRAPH)* 24.4 (1990), 289–298. DOI: 10/dttqmv 88.

[HNPN13] HEITZ, ERIC, NOWROUZEZAHRAI, DEREK, POULIN, PIERRE, and NEYRET, FABRICE. "Filtering Color Mapped Textures and Surfaces". *Proceedings of the Symposium on Interactive 3D Graphics and Games*. Mar. 2013, 129–136. DOI: 10/gfzp5z 89, 92.

[HNPN14] HEITZ, ERIC, NOWROUZEZAHRAI, DEREK, POULIN, PIERRE, and NEYRET, FABRICE. "Filtering Non-Linear Transfer Functions on Surfaces". *IEEE Transactions on Visualization and Computer Graphics* 20.7 (2014), 996–1008. DOI: 10.1109/TVCG.2013.102 89, 92.

[HP02] HEY, HEINRICH and PURGATHOFER, WERNER. "Importance Sampling with Hemispherical Particle Footprints". *Proceedings of the Spring Conference on Computer Graphics (SCCG)*. Apr. 2002, 107–114. DOI: 10/fmx2jp 28.

[HPJ12] HACHISUKA, TOSHIYA, PANTALEONI, JACOPO, and JENSEN, HENRIK WANN. "A Path Space Extension for Robust Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31.6 (Nov. 1, 2012), 191:1–191:10. DOI: 10/gbb6n3 24.

[HSF*25] HUANGFU, QI, SCHORK, LUKAS, FELDMEIER, MICHAEL, GOTTWALD, LEONA, HALL, JULIAN, and GALABOVA, IVET. *HiGHS*. 2025. URL: https://highs.dev 104.

[HSRM25] HERHOLZ, SEBASTIAN, SIK, MARTIN, REICHARDT, LEA, and MANZI, MARCO. "Path Guiding in Production and Recent Advancements". *ACM SIGGRAPH Courses*. 2025. DOI: 10.1145/3721241.3733994 25, 51, 83.

[HWD22] HANIKA, JOHANNES, WEIDLICH, ANDREA, and DROSKE, MARC. "Once-More Scattered Next Event Estimation for Volume Rendering". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 41.4 (2022). DOI: 10.1111/cgf.14583 53, 54, 60, 71.

[HZE*19] Herholz, Sebastian, Zhao, Yangyang, Elek, Oskar, Nowrouzezahrai, Derek, Lensch, Hendrik P. A., and Křivánek, Jaroslav. "Volume Path Guiding Based on Zero-Variance Random Walk Theory". *ACM Transactions on Graphics* 38.3 (June 2019). DOI: 10.1145/3230635 29, 30, 43, 46, 54.

[Ige99] Igehy, Homan. "Tracing Ray Differentials". *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 33. Aug. 1999, 179–186. DOI: 10/c2t9t9 89.

[Int22] Intel. *Intel® Arc™ Graphics Developer Guide for Real-Time Ray Tracing in Games*. 2022. URL: https://www.intel.com/content/www/us/en/developer/articles/guide/real-time-ray-tracing-in-games.html 89.

[JAG18] Jarabo, Adrian, Aliaga, Carlos, and Gutierrez, Diego. "A Radiative Transfer Framework for Spatially-Correlated Materials". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 37.4 (July 2018), 83:1–83:13. DOI: 10/gd52qq 13.

[Jak13] Jakob, Wenzel Alban. "Light Transport on Path-Space Manifolds". PhD thesis. Cornell University, 2013 13, 54, 111.

[JAM*10] Jakob, Wenzel, Arbree, Adam, Moon, Jonathan T., Bala, Kavita, and Marschner, Steve. "A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 29.4 (July 2010), 53:1–53:13. DOI: 10/ftr7st 13.

[Jen95] Jensen, Henrik Wann. "Importance Driven Path Tracing Using the Photon Map". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. 1995, 326–335. DOI: 10/gf2hcr 25, 28.

[Jen96] Jensen, Henrik Wann. "Global Illumination Using Photon Maps". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. Vienna, June 1996, 21–30. DOI: 10/fzc6t9 24.

[JM12] Jakob, Wenzel and Marschner, Steve. "Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 31.4 (July 2012), 58:1–58:13. DOI: 10/gfzq4p 25, 30.

[JMLH01] Jensen, Henrik Wann, Marschner, Stephen R., Levoy, Marc, and Hanrahan, Pat. "A Practical Model for Subsurface Light Transport". *Annual Conference Series (Proceedings of SIGGRAPH)*. ACM Press, Aug. 2001, 511–518. DOI: 10/chdvp7 82.

[JMM*14] Jarabo, Adrian, Marco, Julio, Munoz, Adolfo, Buisan, Raul, Jarosz, Wojciech, and Gutierrez, Diego. "A Framework for Transient Rendering". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 33.6 (Nov. 2014), 177:1–177:10. DOI: 10/gfznb8 54.

[JNSJ11] Jarosz, Wojciech, Nowrouzezahrai, Derek, Sadeghi, Iman, and Jensen, Henrik Wann. "A Comprehensive Theory of Volumetric Radiance Estimation Using Photon Points and Beams". *ACM Transactions on Graphics* 30.1 (Jan. 1, 2011). DOI: 10/fcdh2f 24.

[Jun24] Jung, Alisa. "Mollifying Realistic Image Synthesis for Time Constrained Rendering". PhD thesis. Karlsruhe Institute of Technology, Oct. 2024 13.

[Kaj86] Kajiya, James T. "The Rendering Equation". *Computer Graphics (Proceedings of SIGGRAPH)* 20.4 (Aug. 1986), 143–150. DOI: 10/cvf53j 21.

[KCSG18] Kulla, Christopher, Conty, Alejandro, Stein, Clifford, and Gritz, Larry. "Sony Pictures Imageworks Arnold". *ACM Transactions on Graphics* 37.3 (Aug. 2018), 29:1–29:18. DOI: 10/gfjkn7 88.

[KD13] Kaplanyan, Anton S. and Dachsbacher, Carsten. "Path Space Regularization for Holistic and Robust Light Transport". *Computer Graphics Forum (Proceedings of Eurographics)* 32.2 (2013), 63–72. DOI: 10/gbc3p8 23, 48.

[Kd14] Křivánek, Jaroslav and d'Eon, Eugene. "A Zero-variance-based Sampling Scheme for Monte Carlo Subsurface Scattering". *SIGGRAPH Talks*. 2014, 66:1–66:1. DOI: 10/gfzq7n 54, 81.

[KF12] Kulla, Christopher and Fajardo, Marcos. "Importance Sampling Techniques for Path Tracing in Participating Media". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 31.4 (June 2012), 1519–1528. DOI: 10/f35f4k 21, 53.

[KFF*15] Keller, Alexander, Fascione, Luca, Fajardo, Marcos, Georgiev, Iliyan, Christensen, Per, Hanika, Johannes, Eisenacher, Christian, and Nichols, Greg. "The Path-Tracing Revolution in the Movie Industry". *ACM SIGGRAPH Courses*. 2015, 24:1–24:7. DOI: 10/gfzp5t 88.

[KGH*14] Křivánek, Jaroslav, Georgiev, Iliyan, Hachisuka, Toshiya, Vévoda, Petr, Šik, Martin, Nowrouzezahrai, Derek, and Jarosz, Wojciech. "Unifying points, beams, and paths in volumetric light transport simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014). DOI: 10.1145/2601097.2601219 24, 54.

[KHD14] Kaplanyan, Anton S., Hanika, Johannes, and Dachsbacher, Carsten. "The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014), 102:1–102:13. DOI: 10/f6cz85 25, 30, 89, 110.

[Kis65] Kish, Leslie. *Survey Sampling*. John Wiley & Sons, 1965. ISBN: 978-0-471-48900-9 41.

[KKCF13] King, Alan, Kulla, Christopher, Conty, Alejandro, and Fajardo, Marcos. "BSSRDF Importance Sampling". *ACM SIGGRAPH Talks*. 2013. DOI: 10/gfz5nz 82.

[KLR*23] Kettunen, Markus, Lin, Daqi, Ramamoorthi, Ravi, Bashford-Rogers, Thomas, and Wyman, Chris. "Conditional Resampled Importance Sampling and ReSTIR". *ACM SIGGRAPH Asia Conference Papers*. Dec. 2023. DOI: 10.1145/3610548.3618245 25.

[KMH95] Kolb, Craig, Mitchell, Don, and Hanrahan, Pat. "A Realistic Camera Model for Computer Graphics". *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 29. 1995, 317–324. DOI: 10/dkghpb 13.

[KSAC02] Kelemen, Csaba, Szirmay-Kalos, László, Antal, György, and Csonka, Ferenc. "A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm". *Computer Graphics Forum* 21.3 (Sept. 2002), 531–540. DOI: 10/bfrsqn 25.

[LGXT17] Lee, Mark, Green, Brian, Xie, Feng, and Tabellion, Eric. "Vectorized Production Path Tracing". *Proceedings of High Performance Graphics*. 2017. DOI: 10.1145/3105762.3105768 89.

[LKB*22] Lin, Daqi, Kettunen, Markus, Bitterli, Benedikt, Pantaleoni, Jacopo, Yuksel, Cem, and Wyman, Chris. "Generalized Resampled Importance Sampling: Foundations of ReSTIR". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 2022), 75:1–75:23. DOI: 10/gqjn7b 25, 82.

[LKL*13] Lehtinen, Jaakko, Karras, Tero, Laine, Samuli, Aittala, Miika, Durand, Fredo, and Aila, Timo. "Gradient-Domain Metropolis Light Transport". *ACM Transactions on Graphics* 32.4 (2013), 95:1–95:12. DOI: 10/gbdghd 25, 82.

[LLC*10] Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D.S., Lewis, J.P., Perlin, K., and Zwicker, M. "A Survey of Procedural Noise Functions". *Computer Graphics Forum* 29.8 (2010), 2579–2600. DOI: 10.1111/j.1467-8659.2010.01827.x 90.

[LSD25] Lelyakin, Sergey, Schüßler, Vincent, and Dachsbacher, Carsten. "Sampling of Anisotropic Spatial Gaussians for Path Guiding". *Eurographics - Posters*. May 2025. DOI: 10.2312/egp.20251017 4.

[LW93] Lafortune, Eric P. and Willems, Yves D. "Bi-Directional Path Tracing". *Proceedings of the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*. Vol. 93. Dec. 1993, 145–153 22.

[LW95] Lafortune, Eric P. and Willems, Yves D. "A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. NY, June 1995, 11–20. DOI: 10/gfz5ns 25, 28.

[LWS*25] Li, Beichen, Wu, Rundi, Solar-Lezama, Armando, Zheng, Changxi, Shi, Liang, Bickel, Bernd, and Matusik, Wojciech. "VLMaterial: Procedural Material Generation with Large Vision-Language Models". *The Thirteenth International Conference on Learning Representations*. 2025 90.

[MELC17] Martino, L., Elvira, V., Luengo, D., and Corander, J. "Layered adaptive importance sampling". *Statistics and Computing* 27 (May 2017). DOI: 10.1007/s11222-016-9642-5 80.

[MGN17] Müller, Thomas, Gross, Markus, and Novák, Jan. "Practical Path Guiding for Efficient Light-Transport Simulation". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 36.4 (June 2017), 91–100. DOI: 10/gbnvrs 25, 27, 28, 43.

[MMR*19] Müller, Thomas, McWilliams, Brian, Rousselle, Fabrice, Gross, Markus, and Novák, Jan. "Neural Importance Sampling". *ACM Transactions on Graphics* 38.5 (Oct. 2019), 145:1–145:19. DOI: 10.1145/3341156 29.

[MRYS24] Meyer, Joshua, Rath, Alexander, Yazici, Ömercan, and Slusallek, Philipp. "MARS: Multi-sample Allocation through Russian Roulette and Splitting". *ACM SIGGRAPH Asia Conference Papers*. Dec. 2024. DOI: 10.1145/3680528.3687636 23.

[MT12] Marsden, Jerol E. and Tromba, Anthony. *Vector Calculus*. 6th ed. W. H. Freeman, 2012. ISBN: 978-1-4292-1508-4 9.

[Mül19] Müller, Thomas. ""Practical Path Guiding" in Production". *ACM SIGGRAPH Courses: Path Guiding in Production*. New York, NY, USA, 2019, 18:35–18:48. doi: 10.1145/3305366.3328091 29, 34, 40, 43.

[Mur12] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. isbn: 978-0-262-01802-9 41.

[NVI22] NVIDIA. *Shader Execution Reordering*. 2022. url: https://developer.nvidia.com/blog/improve-shader-performance-and-in-game-frame-rates-with-shader-execution-reordering/ 89.

[OKH*17] Otsu, Hisanari, Kaplanyan, Anton S., Hanika, Johannes, Dachsbacher, Carsten, and Hachisuka, Toshiya. "Fusing State Spaces for Markov Chain Monte Carlo Rendering". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (July 2017), 74:1–74:10. doi: 10/gbxjs9 25, 82, 85.

[Owe13] Owen, Art B. *Monte Carlo Theory, Methods and Examples*. 2013. url: https://artowen.su.domains/mc/ (visited on 09/17/2025) 39.

[Pan17] Pantaleoni, Jacopo. "Charted Metropolis Light Transport". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36.4 (July 2017), 75:1–75:14. doi: 10/gfzq78 25, 82, 85.

[Pea90] Peachey, Darwin. *Texture On Demand*. Technical Memo 217. Pixar, Jan. 1990 89.

[Pet21] Peters, Christoph. "BRDF Importance Sampling for Polygonal Lights". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 40.4 (July 2021), 140:1–140:14. doi: 10/gqjn8c 22.

[PKK00] Pauly, Mark, Kollig, Thomas, and Keller, Alexander. "Metropolis Light Transport for Participating Media". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. 2000, 11–22. doi: 10/gfzm93 6.

[PM00] Pelleg, Dau and Moore, Andrew. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters". *In Proceedings of the 17th International Conf. on Machine Learning*. 2000, 727–734 42.

[PPD98] Paquette, Eric, Poulin, Pierre, and Drettakis, George. "A Light Hierarchy for Fast Rendering of Scenes with Many Lights". *Computer Graphics Forum (Proceedings of Eurographics)* 17.3 (Sept. 1998), C63–C74 22.

[PRDD15] Popov, Stefan, Ramamoorthi, Ravi, Durand, Fredo, and Drettakis, George. "Probabilistic Connections for Bidirectional Path Tracing". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 34.4 (2015), 75–86. doi: 10/gfzwbh 24.

[PVG19] Pediredla, Adithya, Veeraraghavan, Ashok, and Gkioulekas, Ioannis. "Ellipsoidal Path Connections for Time-Gated Rendering". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 38.4 (July 2019). doi: 10/gf5jbm 54.

[PWSF24] Pharr, Matt, Wronski, Bartlomiej, Salvi, Marco, and Fajardo, Marcos. "Filtering after Shading with Stochastic Texture Filtering". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 7.1 (May 2024), 14:1–14:20. doi: 10/gtvf3c 92.

[RGH*20] Rath, Alexander, Grittmann, Pascal, Herholz, Sebastian, Vévoda, Petr, Slusallek, Philipp, and Křivánek, Jaroslav. "Variance-Aware Path Guiding". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). doi: 10/gg8xdb 43, 48.

[RGH*22] Rath, Alexander, Grittmann, Pascal, Herholz, Sebastian, Weier, Philippe, and Slusallek, Philipp. "EARS: Efficiency-Aware Russian Roulette and Splitting". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 22, 2022), 81:1–81:14. issn: 0730-0301. doi: 10/gqjn7j 20, 21.

[RHJD18] Reibold, Florian, Hanika, Johannes, Jung, Alisa, and Dachsbacher, Carsten. "Selective Guided Sampling with Complete Light Transport Paths". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37.6 (Dec. 2018), 223:1–223:14. doi: 10/gf2g93 27, 29, 34–36, 43, 48.

[RHL20] Ruppert, Lukas, Herholz, Sebastian, and Lensch, Hendrik P. A. "Robust Fitting of Parallax-Aware Mixtures for Path Guiding". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). doi: 10/gg8xc6 25, 27, 29, 40, 42, 43.

[RLD*12] Reiner, Tim, Lefebvre, Sylvain, Diener, Lorenz, García, Ismael, Jobard, Bruno, and Dachsbacher, Carsten. "A runtime cache for interactive procedural modeling". *Computers & Graphics* 36.5 (2012), 366–375 90.

[Sch78] Schwarz, Gideon. "Estimating the Dimension of a Model". *The Annals of Statistics* 6.2 (1978), 461–464. doi: 10.1214/aos/1176344136 42.

[SD07] Su, Ting and Dy, Jennifer G. "In Search of Deterministic Methods for Initializing K-Means and Gaussian Mixture Clustering". *Intell. Data Anal.* 11.4 (Dec. 2007), 319–338 42.

[SG31] Smith, T. and Guild, J. "The C.I.E. Colorimetric Standards and Their Use". *Transactions of the Optical Society* 33.3 (Jan. 1931), 73–134. doi: 10/fvzpzn 11.

[SHD24a] Schüßler, Vincent, Hanika, Johannes, and Dachsbacher, Carsten. "Bridge Sampling for Connections via Multiple Scattering Events". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 43.4 (2024). doi: 10.1111/cgf.15160 4.

[SHD24b] Schüßler, Vincent, Hanika, Johannes, and Dachsbacher, Carsten. *Implementation of Bridge Sampling for Connections via Multiple Scattering Events*. June 2024. doi: 10.5281/zenodo.12068529 63.

[SHJD22a] Schüßler, Vincent, Hanika, Johannes, Jung, Alisa, and Dachsbacher, Carsten. *Implementation of Path Guiding with Vertex Triplet Distributions*. July 2022. doi: 10.5281/zenodo.6669884 43.

[SHJD22b] Schüßler, Vincent, Hanika, Johannes, Jung, Alisa, and Dachsbacher, Carsten. "Path Guiding with Vertex Triplet Distributions". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 41.4 (2022). doi: 10.1111/cgf.14582 4.

[SHS*25a] Schüßler, Vincent, Hanika, Johannes, Sauvage, Basile, Dischler, Jean-Michel, and Dachsbacher, Carsten. *Implementation of "'Selective Caching in Procedural Texture Graphs for Path Tracing"'*. June 2025. doi: 10.5281/zenodo.15585724 104.

[SHS*25b] Schüßler, Vincent, Hanika, Johannes, Sauvage, Basile, Dischler, Jean-Michel, and Dachsbacher, Carsten. "Selective Caching in Procedural Texture Graphs for Path Tracing". *Proceedings of the Eurographics Symposium on Rendering (EGSR)*. 2025. doi: 10.2312/sr.20251176 4.

[ŠK18] Šik, Martin and Křivánek, Jaroslav. "Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation". *IEEE Transactions on Visualization and Computer Graphics* 26.4 (2018), 1821–1840. doi: 10.1109/TVCG.2018.2880455 25.

[SLH*20] Shi, Liang, Li, Beichen, Hašan, Miloš, Sunkavalli, Kalyan, Boubekeur, Tamy, Mech, Radomir, and Matusik, Wojciech. "Match: Differentiable Material Graphs for Procedural Material Capture". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39.6 (Nov. 27, 2020), 196:1–196:15. doi: 10/ghnnkm 90.

[SLK*24] Sawhney, Rohan, Lin, Daqi, Kettunen, Markus, Bitterli, Benedikt, Ramamoorthi, Ravi, Wyman, Chris, and Pharr, Matt. "Decorrelating ReSTIR Samplers via MCMC Mutations". *ACM Transactions on Graphics* 43.1 (Jan. 2024). doi: 10.1145/3629166 25, 85.

[SW01] Suykens, Frank and Willems, Yves D. "Path Differentials and Applications". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. 2001, 257–268. doi: 10/bq6pdj 89.

[TH19] Tokuyoshi, Yusuke and Harada, Takahiro. "Hierarchical Russian Roulette for Vertex Connections". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 38.4 (July 2019), 36:1–36:12. doi: 10/gf5jbg 24.

[THD17] Tessari, Lorenzo, Hanika, Johannes, and Dachsbacher, Carsten. "Local Quasi-Monte Carlo Exploration". *Proceedings of EGSR (Experimental Ideas & Implementations)*. 2017. doi: 10.2312/sre20171196 25.

[Uhl95] Uhlmann, Jeffrey. "Dynamic map building and localization: New theoretical foundation". PhD thesis. University of Oxford, 1995 93.

[Vea97] Veach, Eric. "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford University, Dec. 1997 6, 10, 17, 22, 23.

[VG95a] Veach, Eric and Guibas, Leonidas J. "Bidirectional Estimators for Light Transport". *Photorealistic Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. 1995, 145–167. doi: 10/gfznbh 22.

[VG95b] Veach, Eric and Guibas, Leonidas J. "Optimally Combining Sampling Techniques for Monte Carlo Rendering". *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 29. Aug. 1995, 419–428. doi: 10/d7b6n4 18, 79.

[VG97] Veach, Eric and Guibas, Leonidas J. "Metropolis Light Transport". *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 31. Aug. 1997, 65–76. doi: 10/bkjqj4 25.

[VHH*19] Vorba, Jiří, Hanika, Johannes, Herholz, Sebastian, Müller, Thomas, Křivánek, Jaroslav, and Keller, Alexander. "Path Guiding in Production". *ACM SIGGRAPH Courses*. July 2019, 18:1–18:77. DOI: 10.1145/3305366.3328091 25, 27, 54.

[VK16] Vorba, Jiří and Křivánek, Jaroslav. "Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 35.4 (July 2016), 42:1–42:11. DOI: 10/f89mcz 20, 21, 70.

[VKK18] Vévoda, Petr, Kondapaneni, Ivo, and Křivánek, Jaroslav. "Bayesian Online Regression for Adaptive Direct Illumination Sampling". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 37.4 (July 2018), 125:1–125:12. DOI: 10/gd52ss 61.

[VKŠ*14] Vorba, Jiří, Karlík, Ondřej, Šik, Martin, Ritschel, Tobias, and Křivánek, Jaroslav. "On-Line Learning of Parametric Mixture Models for Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (Aug. 2014), 101:1–101:11. DOI: 10/f6c2cp 25, 27, 28, 32, 40, 42, 45.

[VWF18] Villemin, Ryusuke, Wrenninge, Magnus, and Fong, Julian. "Efficient Unbiased Rendering of Thin Participating Media". *Journal of Computer Graphics Techniques (JCGT)* 7.3 (Sept. 2018), 50–65. ISSN: 2331-7418. URL: http://jcgt.org/published/0007/03/03/ 20, 73.

[WDH*21] Weier, Philippe, Droske, Marc, Hanika, Johannes, Weidlich, Andrea, and Vorba, Jiří. "Optimised Path Space Regularisation". *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 40.4 (2021). DOI: 10.1111/cgf.14347 23, 48.

[WGGH20] West, Rex, Georgiev, Iliyan, Gruson, Adrien, and Hachisuka, Toshiya. "Continuous Multiple Importance Sampling". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: 10/gg8xcr 25, 77, 78.

[WHD17] Weber, Pascal, Hanika, Johannes, and Dachsbacher, Carsten. "Multiple Vertex Next Event Estimation for Lighting in Dense, Forward-Scattering Media". *Computer Graphics Forum (Proceedings of Eurographics)* 36.2 (Apr. 2017), 21–30. DOI: 10/gbm2gf 53.

[Wil83] Williams, Lance. "Pyramidal Parametrics". *Computer Graphics (Proceedings of SIGGRAPH)* 17.3 (July 1, 1983), 1–11. DOI: 10/cq4xrd 89.

[WM24] West, Rex and Mukherjee, Sayan. "Stylized Rendering as a Function of Expectation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 43.4 (July 2024), 96:1–96:19. DOI: 10/gt48vh 92.

[WND*14] Wilkie, Alexander, Nawaz, Sehera, Droske, Marc, Weidlich, Andrea, and Hanika, Johannes. "Hero Wavelength Spectral Sampling". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 33.4 (June 2014), 123–131. DOI: 10/f6fgb4 11.

[WSD24] Werner, Mirco, Schüßler, Vincent, and Dachsbacher, Carsten. "ReSTIR Subsurface Scattering for Real-Time Path Tracing". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 7.3 (Aug. 2024). DOI: 10.1145/3675372 4, 81–83.

[WW12] Wilkie, Alexander and Weidlich, Andrea. "Polarised Light in Computer Graphics". *ACM SIGGRAPH Asia Courses*. 2012. DOI: 10/gfz5nn 13.

[YB18] Yang, Y. and Barnes, C. "Approximate Program Smoothing Using Mean-Variance Statistics, with Application to Procedural Shader Bandlimiting". *Computer Graphics Forum (Proceedings of Eurographics)* 37.2 (2018), 443–454. DOI: 10/gd2jb7 89, 92.

[ZD20] Zirr, Tobias and Dachsbacher, Carsten. "Path Differential-Informed Stratified MCMC and Adaptive Forward Path Sampling". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39.6 (Nov. 27, 2020), 246:1–246:19. DOI: 10/gsmm9d 89.

[ZGJ20] Zeltner, Tizian, Georgiev, Iliyan, and Jakob, Wenzel. "Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). DOI: 10/gg8xc8 23.

[ZHD18] Zirr, Tobias, Hanika, Johannes, and Dachsbacher, Carsten. "Re-Weighting Firefly Samples for Improved Finite-Sample Monte Carlo Estimates". *Computer Graphics Forum* 37.6 (Sept. 1, 2018), 410–421. DOI: 10/gdv89k 34.

[ZXS*21] Zhu, Shilin, Xu, Zexiang, Sun, Tiancheng, Kuznetsov, Alexandr, Meyer, Mark, Jensen, Henrik Wann, Su, Hao, and Ramamoorthi, Ravi. "Hierarchical Neural Reconstruction for Path Guiding Using Hybrid Path and Photon Samples". *ACM Transactions on Graphics* 40.4 (July 2021). DOI: 10.1145/3450626.3459810 28, 32.

[ZZ19] Zheng, Quan and Zwicker, Matthias. "Learning to Importance Sample in Primary Sample Space". *Computer Graphics Forum* 38.2 (2019), 169–179. DOI: 10/ggkjx5 29.