DOI: 10.1111/cgf.14582
Eurographics Symposium on Rendering 2022
A. Ghosh and L.-Y. Wei
(Guest Editors)

Volume 41 (2022), Number 4

Path Guiding with Vertex Triplet Distributions

Vincent Schüßler, Johannes Hanika, Alisa Jung and Carsten Dachsbacher

Karlsruhe Institute of Technology, Germany

Abstract

Good importance sampling strategies are decisive for the quality and robustness of photorealistic image synthesis with Monte Carlo integration. Path guiding approaches use transport paths sampled by an existing base sampler to build and refine a guiding distribution. This distribution then guides subsequent paths in regions that are otherwise hard to sample. 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 9D guiding distribution over vertex triplets that adapts to the full measurement contribution with a 9D Gaussian mixture model (GMM). For incremental path sampling, we query the model for the last two vertices of a path prefix, resulting in a 3D conditional distribution with which we sample the next vertex along the path. To make this approach scalable, we partition the scene with an octree and learn a local GMM for each leaf separately. In a learning phase, we sample paths using the current guiding distribution and collect triplets of path vertices. We resample these triplets online and keep only a fixed-size subset in reservoirs. After each progression, we obtain new GMMs from triplet samples by an initial hard clustering followed by expectation maximization. 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 a light tracer. Therefore our method can guide both a path tracer and light tracer from a jointly learned guiding distribution.

CCS Concepts

• ***Computing methodologies*** → *Ray tracing;*

1. Introduction

Monte Carlo light transport has become the standard technique for realistic offline rendering. Good importance sampling strategies to keep variance low are crucial for its efficiency. 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] where path sampling techniques adapt to the scene using information (e.g. incident illumination on surfaces) gathered from previously sampled paths. 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 [GKDS12]), or Markov chains [Has70]. Guiding can thus often provide an unbiased, or more temporally stable alternative.

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. And 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 which 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 which 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 our paper makes the following contributions:

- a general, 9D, bidirectional model of the transport operator including both surface and volume transport (section 3 and 4),
- bidirectional guided path construction using this model (section 5),
- effective training of the model using reservoir-based resampling and a clustering approach based on principal component analysis (PCA) (section 6).

2. Background and related work

Our goal in rendering is to compute the path integral [Vea97] for every pixel j:

$$I_j = \int_{\mathcal{P}} f_j(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}, \tag{1}$$

where $\mathcal{P}$ is the path space and $\bar{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}) \in \mathcal{P}_k \subset \mathcal{P}$ is a path of length k. The measurement contribution function (MCF) f_j is defined as

$$\begin{aligned} f_j(\bar{\mathbf{x}}) = {} & W_j(\mathbf{x}_0, \mathbf{x}_1)\, L_e(\mathbf{x}_{k-1}, \mathbf{x}_{k-2}) G(\mathbf{x}_0, \mathbf{x}_1)\, T(\mathbf{x}_0, \mathbf{x}_1) \\ & \cdot \prod_{i=1}^{k-2} f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})\, G(\mathbf{x}_i, \mathbf{x}_{i+1})\, T(\mathbf{x}_i, \mathbf{x}_{i+1}), \end{aligned} \tag{2}$$

$$G(\mathbf{x}, \mathbf{y}) = \frac{D(\mathbf{x}, \mathbf{y})\, D(\mathbf{y}, \mathbf{x})}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{3}$$

$$D(\mathbf{x}, \mathbf{y}) = \begin{cases} |n(\mathbf{x}) \cdot \omega_{\mathbf{x} \to \mathbf{y}}| & \text{if } \mathbf{x} \text{ is on a surface,} \\ 1 & \text{if } \mathbf{x} \text{ is in a medium,} \end{cases} \tag{4}$$

where W_j is the response of pixel j, L_e is emitted radiance, T is transmittance, f_s is the BSDF or the phase function multiplied by the scattering coefficient μ_s, $n(\mathbf{x})$ is the surface normal, and $\omega_{\mathbf{x} \to \mathbf{y}}$ is the normalized direction from $\mathbf{x}$ to $\mathbf{y}$.

Monte Carlo integration is the common method to compute the path integral, partly because discontinuities and high dimensionality make other approaches less appealing. Unfortunately, its efficiency strongly depends on available importance sampling strategies. Usually, no single sampling strategy has low variance across the whole path space. Combination of different strategies using MIS [VG95] is essential for robustness. However, in absence of a good strategy, Monte Carlo often undersamples important regions of the path space. Since it cannot explore them reliably, this manifests in bright outliers and high variance.

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 exploring 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 artefacts on the borders of spatial regions of their SD-trees. They propose a 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 using 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, efficient to store, support textured parameters and still support the 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 [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.

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. 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 3.1), before defining the GMM and giving an intuition on how to sample it (section 3.2). Further, we sketch how learning works and how it integrates into a renderer (section 3.3).

In the subsequent sections, we discuss the complete algorithm: details on the guiding distribution (section 4), its sampling (section 5) and learning (section 6). We show an overview of all the individual steps in fig. 1.

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 $B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ which only depend on three consecutive vertices $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}$, i.e.

$$B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) T(\mathbf{x}_i, \mathbf{x}_{i+1}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \tag{5}$$

and we will use the shorthand $B_i = B(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ indexed by the centre 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

$$f(\bar{\mathbf{x}}) = W(\mathbf{x}_0) \cdot G(\mathbf{x}_0, \mathbf{x}_1) T(\mathbf{x}_0, \mathbf{x}_1) \prod_{i=1}^{k-2} B_i \cdot L_e(\mathbf{x}_{k-1}). \tag{6}$$

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

$$p(\mathbf{x}_{i+1}|\mathbf{x}_i, \mathbf{x}_{i-1}) \sim B_i \cdot \int \prod_{j=i+1}^{k-2} B_j \cdot L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+2..k-1}. \tag{7}$$

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 $j > i+1$:

$$\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \int f(\bar{\mathbf{x}}) \, \mathrm{d}\{\mathbf{x}_{j<i-1} \mathbf{x}_{j>i+1}\}. \tag{8}$$

Similarly we can define

$$\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i) = \int \mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) \, \mathrm{d}\mathbf{x}_{i+1} \tag{9}$$

$$= \int f(\bar{\mathbf{x}}) \, \mathrm{d}\{\mathbf{x}_{j<i-1} \mathbf{x}_{j>i}\}, \tag{10}$$

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

$$\mathcal{G}(\mathbf{x}_{i+1}|\mathbf{x}_{i-1}, \mathbf{x}_i) = \frac{\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})}{\mathcal{G}(\mathbf{x}_{i-1}, \mathbf{x}_i)}. \tag{11}$$

To analyse 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. (8) and (10) 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, consider the minimal example

$$I_1 = \int B_1 \cdot B_2 \cdot \mathrm{d}\mathbf{x}_{i=0..3} \tag{12}$$

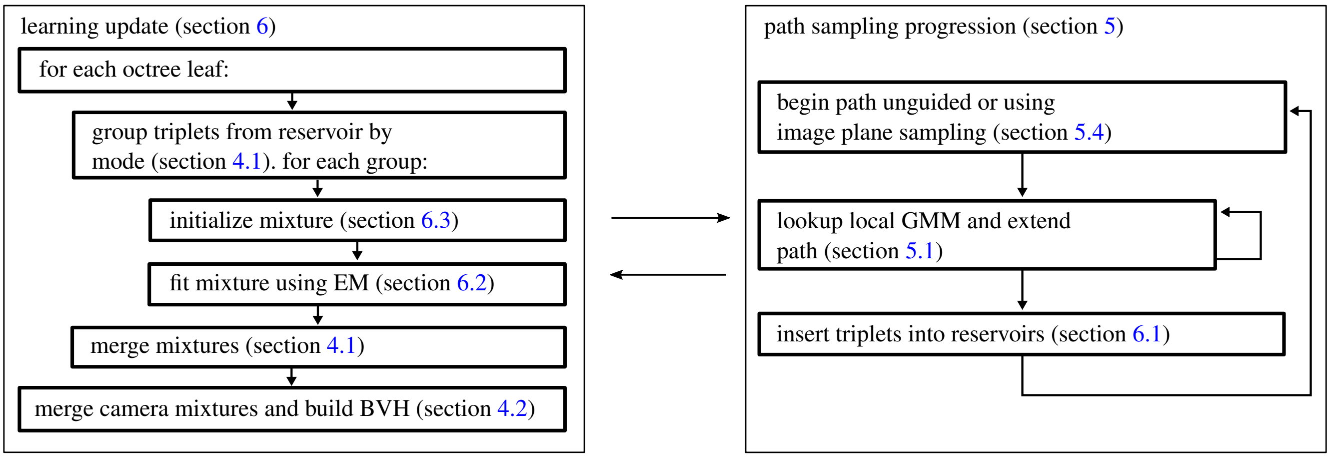

Figure 1: *Overview of the individual steps in our algorithm. During the learning phase, we alternate between learning (left) and path sampling (right).*

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:

$$I_2(\mathbf{x}_1, \mathbf{x}_2) = \int B_1 \cdot B_2 \cdot \mathrm{d}\mathbf{x}_0 \mathrm{d}\mathbf{x}_3 \tag{13}$$

$$= \int B_1 \mathrm{d}\mathbf{x}_0 \cdot \int B_2 \mathrm{d}\mathbf{x}_3. \tag{14}$$

This allows us to separate both eq. (8) and eq. (10) in the numerator and denominator of eq. (11) into two integrals. Figure 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:

$$\mathcal{G}(\mathbf{x}_{i+1}|\mathbf{x}_{i-1}, \mathbf{x}_i) = \frac{\cancel{\int WTG \prod_{j=1}^{i-1} B_j \mathrm{d}\mathbf{x}_{1..i-2}} \cdot B_i \cdot \int \prod_{j=i+1}^{k-2} B_j L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+2..k-1}}{\cancel{\int WTG \prod_{j=1}^{i-1} B_j \mathrm{d}\mathbf{x}_{1..i-2}} \cdot \int \prod_{j=i}^{k-2} B_j L_e(\mathbf{x}_{k-1}) \mathrm{d}\mathbf{x}_{i+1..k-1}} \tag{15}$$

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. (7).

Note that we do not explicitly need to fit and store a GMM for eq. (10), 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. (7). 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 learn sampling

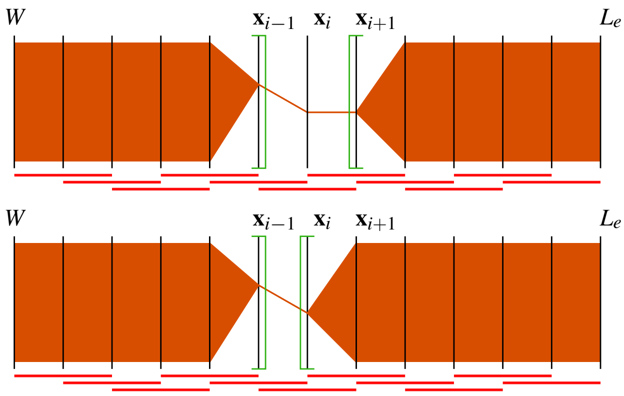

Figure 2: *Parallel coordinates visualisation of path space. Separability of the path space integral (illustrated in orange) can be facilitated by holding consecutive vertices fixed. Since the BSDF f_s depends on three consecutive vertices (indicated as red bars), we need to keep two fixed to be able to separate into two integrals. Top: numerator, bottom: denominator of the conditional in eq. (11). 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 towards the light does not.*

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].

3.2. 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 $\mathcal{N}(\mathbf{t} \mid \mu^k, \Sigma^k)$:

$$p(\mathbf{t}) = \sum_{k=1}^{K} \pi^k \mathcal{N}(\mathbf{t} \mid \mu^k, \Sigma^k), \tag{16}$$

$$\mu^k = (\mu_1^k, \mu_2^k, \mu_3^k) \in \mathbb{R}^9, \quad \Sigma^k = \begin{pmatrix} \Sigma_{11}^k & \Sigma_{12}^k & \Sigma_{13}^k \\ \Sigma_{21}^k & \Sigma_{22}^k & \Sigma_{23}^k \\ \Sigma_{31}^k & \Sigma_{32}^k & \Sigma_{33}^k \end{pmatrix} \in \mathbb{R}^{9\times 9}, \tag{17}$$

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

Guided ~~Sampling~~ 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 $\mathbf{x}_2$:

$$\begin{aligned} &p(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2) \\ &= \sum_{k=1}^{K} \frac{\pi^k \mathcal{N}(\mathbf{x}_1, \mathbf{x}_2 \mid \mu_{1,2}^k, \Sigma_{1,2}^k)}{\sum_{l=1}^{K} \pi^l \mathcal{N}(\mathbf{x}_1, \mathbf{x}_2 \mid \mu_{1,2}^l, \Sigma_{1,2}^l)} \mathcal{N}(\mathbf{x}_3 \mid \mu_{3|1,2}^k, \Sigma_{3|1,2}^k) \\ &=: \sum_{k=1}^{K} \bar{\pi}^k \mathcal{N}(\mathbf{x}_3 \mid \bar{\mu}^k, \bar{\Sigma}^k). \end{aligned} \tag{18}$$

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 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 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 6.1.

3.3. Algorithm

Our algorithm has two distinct phases: learning and rendering. We show an overview of all the individual steps in fig. 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 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 of 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 6.

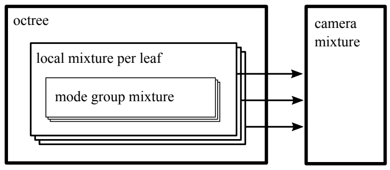

Figure 3: *In our guiding distribution, each leaf of the octree contains a local mixture used for sampling. These are constructed during learning by merging individually fitted mode group mixtures. The camera mixture is constructed for each leaf independently and then merged into the global one.*

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 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 4.2). We show an overview of the different mixtures in our method in fig. 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 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

$$\sum_{k=1}^{K_g} \tilde{\pi}_g^k = 1. \tag{19}$$

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:

$$\pi_g^k = \tilde{\pi}_g^k \cdot \frac{\sum_{i \in I_g} w_i}{\sum_{i=1}^{N} w_i}. \tag{20}$$

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.

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 like 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].

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.

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 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 5.1). Then we sample the position of, or direction to, the next vertex using the chosen component. Only in (near) specular cases (section 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.

5.1. Sampling the conditional GMM

To extend a path prefix $(\ldots, \mathbf{x}_{i-1}, \mathbf{x}_i)$ 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 equation (18):

$$p(\mathbf{x}_{i+1} \mid \mathbf{x}_{i-1}, \mathbf{x}_i) = \sum_{k=1}^{K} \bar{\pi}^k \mathcal{N}(\mathbf{x}_{i+1} \mid \bar{\mu}^k, \bar{\Sigma}^k). \tag{21}$$

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 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 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.

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 $\bar{\mu}$.

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 unfavourable or fail. Instead, we combine with BSDF or phase function sampling at $\mathbf{x}_i$ based on its roughness

$$r := \begin{cases} \sqrt{\alpha} & \text{on microfacet surfaces with roughness } \alpha, \\ \sqrt{1-|g|} & \text{in media with mean cosine } g. \end{cases} \quad (22)$$

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

$$P_{\mathrm{BSDF}}(r) = \max\left(0, 1 - \frac{r}{r_{\max}}\right). \quad (23)$$

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

$$(1 - P_{\mathrm{unguided}}) \cdot P_{\mathrm{BSDF}}(r), \quad (24)$$

and thus the probability for guided position sampling is

$$(1 - P_{\mathrm{unguided}}) \cdot (1 - P_{\mathrm{BSDF}}(r)). \quad (25)$$

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 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.

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_{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 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.

6. Learning a vertex triplet distribution with

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 6.1. We review the EM algorithm for the case of GMMs in section 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 6.3.

Triplet collection and fitting of mixtures is 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 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.

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:

$$\mathbb{E}[\hat{w}_i] = w_i, \quad (26)$$

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

$$\sum_{i \in S} \hat{w}_i = \sum_{i \in I} w_i. \quad (27)$$

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, our target density is proportional to the measurement contribution $f(\bar{\mathbf{x}})$. It follows that the normalized target density is

$$p_{\text{target}}(\bar{\mathbf{x}}) = \frac{1}{Z} f(\bar{\mathbf{x}}), \quad Z := \int_{\mathcal{P}} f(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}. \tag{28}$$

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

$$\mathbb{E}_{\text{target}}[s(\bar{\mathbf{x}})] = \int_{\mathcal{P}} s(\bar{\mathbf{x}})\, p_{\text{target}}(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}. \tag{29}$$

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

$$\mathbb{E}_{\text{target}}[s(\bar{\mathbf{x}})] = \mathbb{E}_{\text{current}}\left[s(\bar{\mathbf{x}}) \cdot \frac{p_{\text{target}}(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})} \right]. \tag{30}$$

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, Chapter 9]:

$$\tilde{F}_n = \frac{\sum_{i=1}^{n} w(\bar{\mathbf{x}}_i)\, s(\bar{\mathbf{x}}_i)}{\sum_{i=1}^{n} w(\bar{\mathbf{x}}_i)}, \tag{31}$$

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

$$w(\bar{\mathbf{x}}) := \frac{f(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})} = Z \cdot \frac{p_{\text{target}}(\bar{\mathbf{x}})}{p_{\text{current}}(\bar{\mathbf{x}})}. \tag{32}$$

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:

$$T = \frac{C}{L}. \tag{33}$$

Before constructing mixtures in each leaf node, we collapse all nodes that received less 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.

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

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} w_i \log p(x_i|\theta), \tag{34}$$

where we abbreviated the GMM parameters as

$$\theta = (\pi^1, \mu^1, \Sigma^1, \dots, \pi^K, \mu^K, \Sigma^K). \tag{35}$$

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 modelling 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 θ_t:

$$r_{ik} = \frac{\pi_t^k \mathcal{N}(x_i \mid \mu_t^k, \Sigma_t^k)}{\sum_{j=1}^{K} \pi_t^j \mathcal{N}(x_i \mid \mu_t^j, \Sigma_t^j)}. \tag{36}$$

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

$$\pi_{t+1}^k = \frac{\sum_{i=1}^{N} v_{ik}}{\sum_{i=1}^{N} w_i}, \ \mu_{t+1}^k = \frac{\sum_{i=1}^{N} v_{ik} x_i}{\sum_{i=1}^{N} v_{ik}}, \ \Sigma_{t+1}^k = \frac{\sum_{i=1}^{N} v_{ik} x_i x_i^T}{\sum_{i=1}^{N} v_{ik}}. \tag{37}$$

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 ($\varepsilon = 10^{-3}$) on the relative increase in log-likelihood:

$$\frac{|\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)|}{|\mathcal{L}(\theta_t)|} < \varepsilon. \tag{38}$$

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 less free parameters Σ_{reg} and use a convex combination of this with the sample covariance Σ [Mur12, Chapter 4.6]:

$$\tilde{\Sigma} = \lambda \Sigma_{\text{reg}} + (1 - \lambda)\Sigma. \tag{39}$$

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

$$\Sigma_{\text{iso}} = r^2 I_9. \tag{40}$$

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

$$\Sigma_{\text{block}} = \left(\frac{Tr(\Sigma_{ij})}{3} \cdot I_3\right)_{i,j=1,2,3} \in \mathbb{R}^{9\times 9}, \tag{41}$$

and combine both into

$$\Sigma_{\text{reg}} = \lambda_{\text{block}}\,\Sigma_{\text{block}} + (1-\lambda_{\text{block}})\,\Sigma_{\text{iso}}. \tag{42}$$

We base the choices of λ and λ_{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:

$$\hat{n} = \frac{\left(\sum_{i=1}^{N} v_i\right)^2}{\sum_{i=1}^{N} v_i^2}. \tag{43}$$

Then we set

$$\lambda = \frac{\hat{n}-1}{(\hat{n}-1)+9}, \quad \lambda_{\text{block}} = 1 - \min\left\{\frac{\hat{n}-1}{3}, 1\right\}. \tag{44}$$

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

6.3. Initial mixture

The results of EM in general depend strongly on the initial parameters of the mixture θ_0. Particularly 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 in 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 to evaluate 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 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 to try 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. $K_{\text{max}} = 128$) 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]

$$\text{AIC}(k) = 2k - 2n\mathcal{L}(\theta(k)). \tag{45}$$

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.

7. Results

We implemented our method in a rendering system and provide source code online [SHJD22].

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.

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.

Colour noise Our methods and SPG are implemented in a spectral rendering system, which produces coloured 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.

7.2. Comparisons

Figure 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 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 SDDM. Therefore, we could only produce a result in half resolution.

Figure 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. 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 did leave the learning phase running for the whole render. This gave better results but still includes bright firefly samples.

Figure 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. 8 we show results on the pool scene, originally presented by Vorba et al. [VKŠ*14]. The 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 longer time for learning. Additionally, light tracing cannot connect to the camera through the dielectric water surface, and thus does mostly not help with learning.

We compare memory usage of our method in table 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 in the same order as previous work.

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.

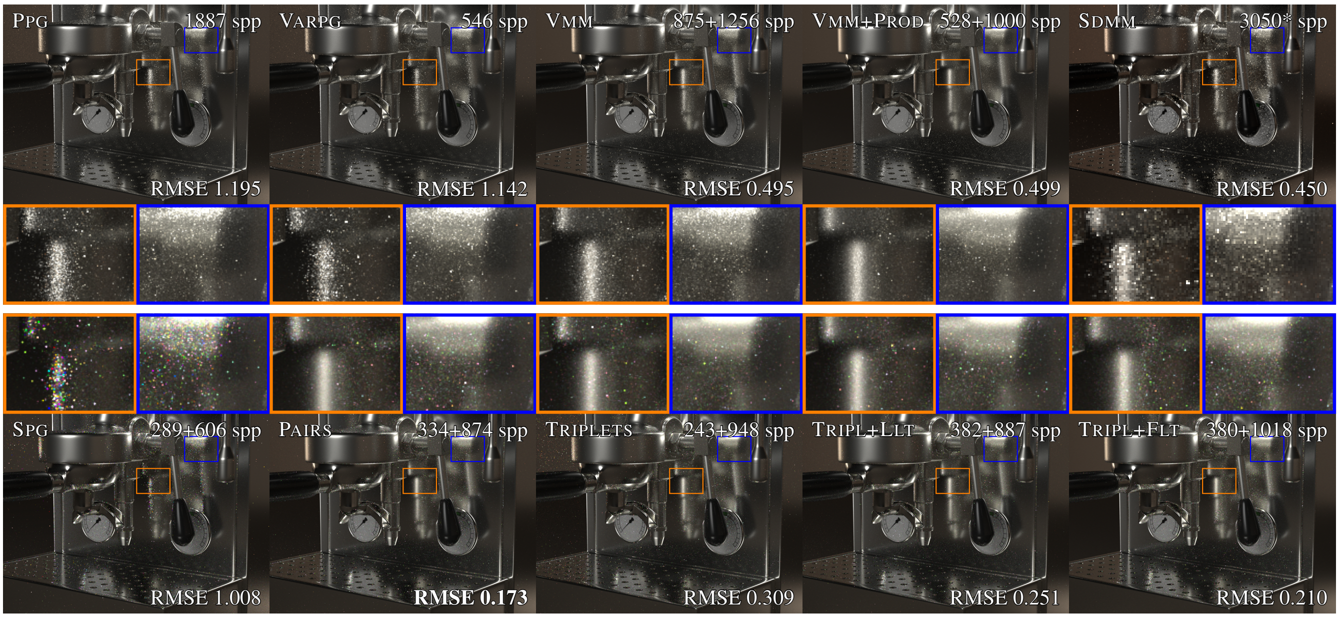

Figure 4: *Glossy surfaces lit by directional emitters produces hard to resolve reflections. Here, sampling the product of BSDF and incident radiance well is important, as can be seen in the difference between* VMM *and* VMM+PROD*, which achieves overall good results here.* PAIRS *show overall better 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 in half resolution, therefore the spp number is much higher than with the other methods.*

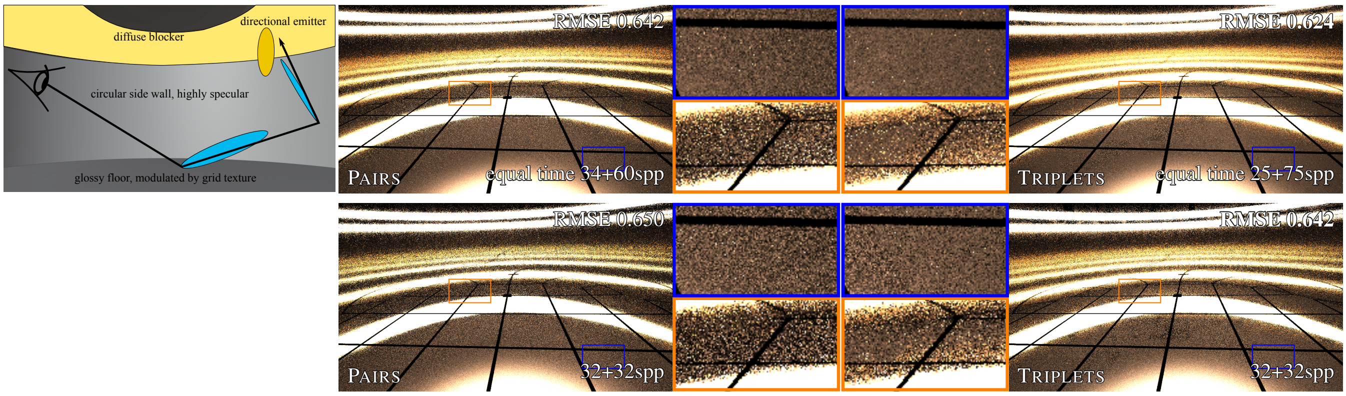

Figure 5: *A circular mirror chamber with a ring-shaped directional emitter on the ceiling. The ground floor has a glossy metal BRDF and is modulated by a grid texture for orientation. This simple scene shows the shortcomings of the GMM using pairs of vertices: it cannot distinguish the incident radiance 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 sub-optimal 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.*

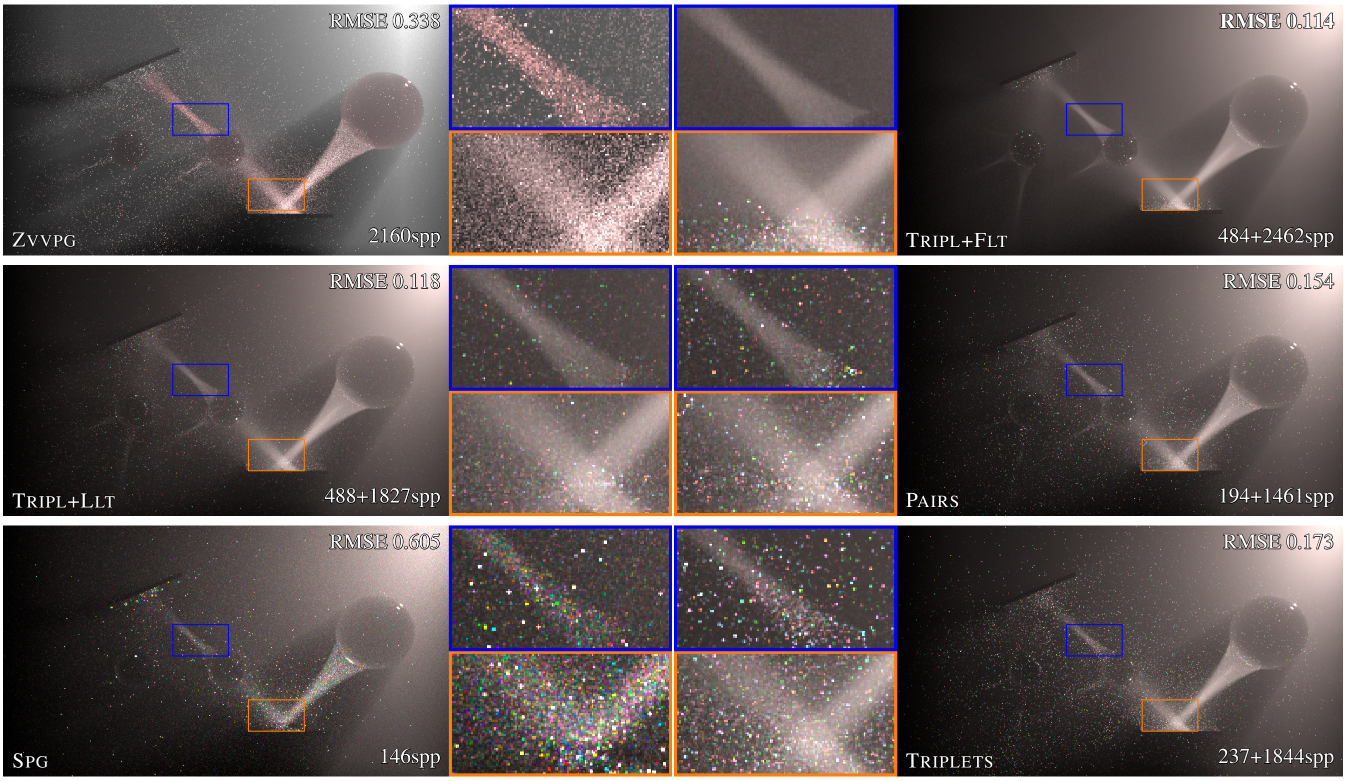

Figure 6: *Spheres and volume caustics by Herholz et al. [HZE*19]. Methods without light tracing (*PAIRS, TRIPLETS, SPG*), take a long time to learn dimmer caustics in the left part.* PAIRS *is surprisingly slow in this scene. We hypothesize that this is due to our implementation not terminating long paths with little contribution using Russian Roulette.*

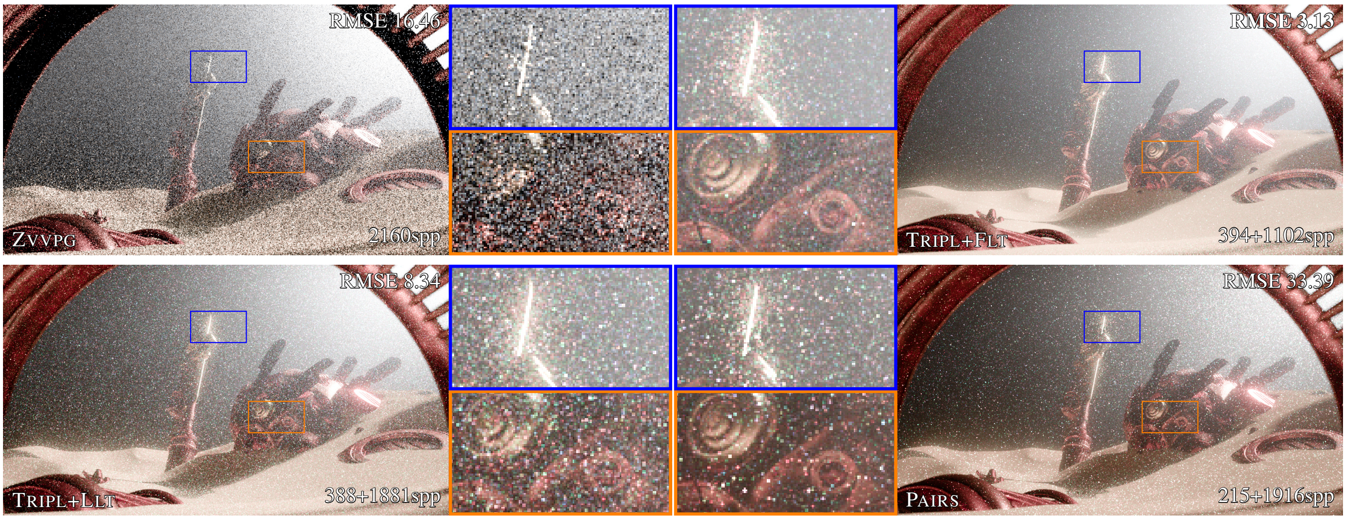

Figure 7: *A hazy scene lit by a directional emitter with mean cosine* $g = 0.98$ *in the homogeneous medium. Here, light tracing contributes important information to the learning as well as the rendering. Learning with* PAIRS *is faster, so in this equal time comparison it achieves better results than learning with* TRIPLETS *as long as light tracing is not used.*

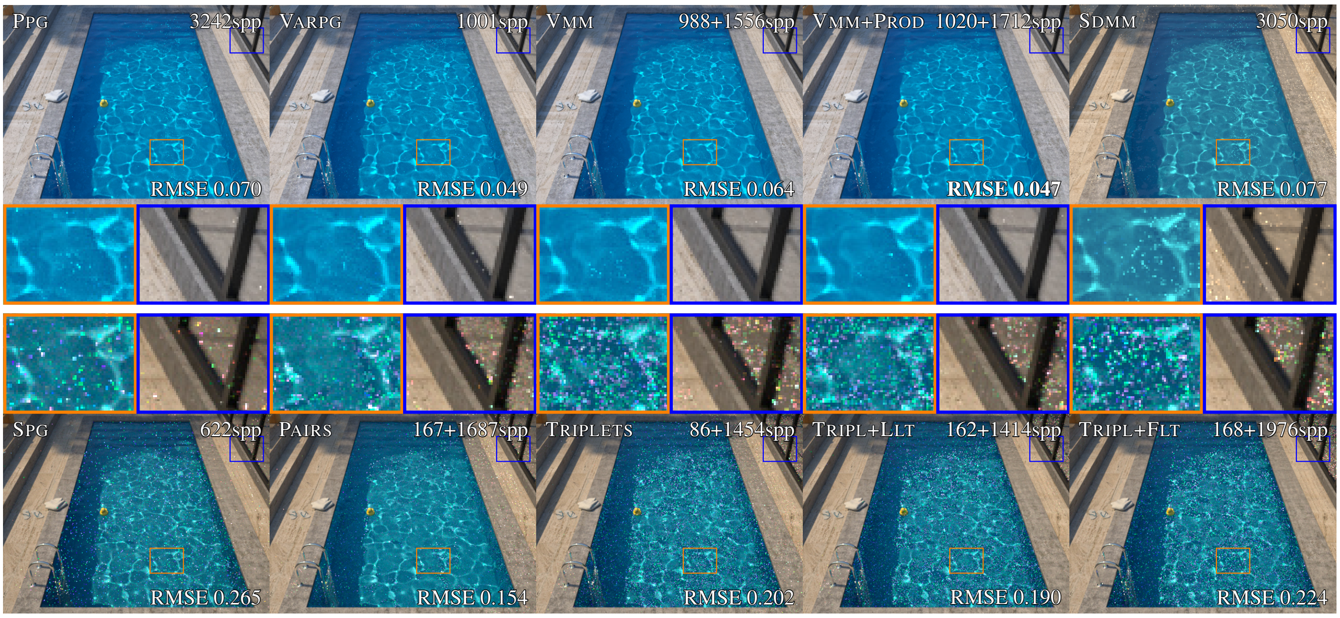

Figure 8: *The pool scene presents a simple case where for each 3D point only a 2D incoming radiance field has to be found. The bottom of the pool is diffuse, so there is nothing to be gained from a more complex approach and training a higher dimensional model just generates overhead both in terms of raw performance as well as convergence of the model itself.*

Table 1: *Our method uses a comparable amount of memory as previous work, shown here for the* POOL *(fig. 8) and* ESPRESSO *(fig. 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

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 connect connections (during learning) using path space regularization [KD13]. Previous work showed how this can benefit path guiding [WDH*21] by discovering features more easily.

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 no redundancy. The result is a 9D mixture model which 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 between 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 which 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 which is able to learn and render faster. We expect future improvements to our model are possible especially in the learning phase as discussed in this work.

Acknowledgements

We thank the following persons for providing models and scenes: Blend Swap user galingong for the espresso machine (fig. 4), Sebastian Herholz for the volume caustic (fig. 6), Pablo Vazquez for "Nishita Sky Demo" (fig. 7) and Michal Timo and Ondřej Karlík for the pool scene (fig. 8). This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 431478017. Open Access funding enabled and organized by Projekt DEAL.

References

[Aka73] Akaike, H. "Information Theory and an Extension of the Maximum Likelihood Principle". *Second International Symposium on Information Theory.* Akademia Kiado. 1973, 267–281 9.

[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 2.

[CDK*09] Cohen, Edith, Duffield, Nick, Kaplan, Haim, et al. "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 7.

[DGJ*20] Diolatzis, Stavros, Gruson, Adrien, Jakob, Wenzel, et al. "Practical Product Path Guiding Using Linearly Transformed Cosines". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* (2020). doi: 10.1111/cgf.14051 3.

[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 8.

[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 2, 3, 9, 10.

[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 3.

[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 2.

[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 1.

[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 1.

[HEV*16] Herholz, Sebastian, Elek, Oskar, Vorba, Jiří, et al. "Product Importance Sampling for Light Transport Path Guiding". *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* (2016). doi: 10/f842dt 3.

[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 2.

[HZE*19] Herholz, Sebastian, Zhao, Yangyang, Elek, Oskar, et al. "Volume Path Guiding Based on Zero-Variance Random Walk Theory". *ACM Transactions on Graphics* 38.3 (June 2019). doi: 10.1145/3230635 3, 10, 12.

[Jen95] Jensen, Henrik Wann. "Importance Driven Path Tracing Using the Photon Map". *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*. Springer-Verlag, 1995, 326–335. doi: 10/gf2hcr 2.

[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 3.

[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 13.

[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 3.

[Kis65] Kish, Leslie. *Survey Sampling*. John Wiley & Sons, 1965 9.

[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: Springer-Verlag, June 1995, 11–20. doi: 10/gfz5ns 2.

[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 1, 2, 10.

[MMR*19] Müller, Thomas, McWilliams, Brian, Rousselle, Fabrice, et al. "Neural Importance Sampling". *ACM Trans. Graph.* 38.5 (Oct. 2019), 145:1–145:19. doi: 10.1145/3341156 2.

[Mül19] Müller, Thomas. ""Practical Path Guiding" in Production". *ACM SIGGRAPH Courses: Path Guiding in Production*. New York, NY, USA: ACM, 2019, 18:35–18:48. doi: 10.1145/3305366.3328091 2, 5, 8, 10.

[Mur12] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012 8.

[Owe13] Owen, Art B. *Monte Carlo Theory, Methods and Examples*. To be published, 2013. url: https://statweb.stanford.edu/~owen/mc/ (visited on 06/07/2019) 8.

[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*. Morgan Kaufmann, 2000, 727–734 9.

[RGH*20] Rath, Alexander, Grittmann, Pascal, Herholz, Sebastian, et al. "Variance-Aware Path Guiding". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 8, 2020). doi: 10/gg8xdb 10, 13.

[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 1–3, 5, 6, 10, 13.

[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 1–3, 8–10.

[Sch78] SCHWARZ, GIDEON. "Estimating the Dimension of a Model". *The Annals of Statistics* 6.2 (1978), 461–464. DOI: 10.1214/aos/1176344136 9.

[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 9.

[SHJD22] SCHÜSSLER, VINCENT, HANIKA, JOHANNES, JUNG, ALISA, and DACHSBACHER, CARSTEN. *Implementation of Path Guiding with Vertex Triplet Distributions*. July 2022. DOI: 10.5281/zenodo.6669884 9.

[Vea97] VEACH, ERIC. "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford University, Dec. 1997 2.

[VG95] VEACH, ERIC and GUIBAS, LEONIDAS J. "Optimally Combining Sampling Techniques for Monte Carlo Rendering". *Annual Conference Series (Proceedings of SIGGRAPH)*. Vol. 29. ACM Press, Aug. 1995, 419–428. DOI: 10/d7b6n4 2.

[VHH*19] VORBA, JIŘÍ, HANIKA, JOHANNES, HERHOLZ, SEBASTIAN, et al. "Path Guiding in Production". *ACM SIGGRAPH Courses*. July 2019, 18:1–18:77. DOI: 10.1145/3305366.3328091 1.

[VKŠ*14] VORBA, JIŘÍ, KARLÍK, ONDŘEJ, ŠIK, MARTIN, et al. "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 1, 2, 4, 8–10.

[WDH*21] WEIER, PHILIPPE, DROSKE, MARC, HANIKA, JOHANNES, et al. "Optimised Path Space Regularisation". *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 40.4 (2021). DOI: 10.1111/cgf.14347 13.

[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 5.

[ZXS*21] ZHU, SHILIN, XU, ZEXIANG, SUN, TIANCHENG, et al. "Hierarchical Neural Reconstruction for Path Guiding Using Hybrid Path and Photon Samples". *ACM Trans. Graph.* 40.4 (July 2021). DOI: 10.1145/3450626.3459810 2, 4.

[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 2.

Appendix A: Conditional and marginal normal distributions

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

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \tag{46}$$

the marginal distribution of x_2 is simply

$$x_2 \sim \mathcal{N}(\mu_2, \Sigma_{22}). \tag{47}$$

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

$$x_{2|1} \sim \mathcal{N}(\mu_{2|1}, \Sigma_{2|1}) \tag{48}$$

using

$$\begin{aligned} \bar{\mu}_{2|1} &= \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \\ \bar{\Sigma}_{2|1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}. \end{aligned} \tag{49}$$

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