Path Guiding in Production

Jiří Vorba (Organizer)
Weta Digital

Johannes Hanika (Organizer)
Weta Digital

Sebastian Herholz
University of Tübingen

Thomas Müller
NVIDIA[1]

Jaroslav Křivánek
Charles University, Prague
Render Legion

Alexander Keller
NVIDIA

SIGGRAPH 2019 Course

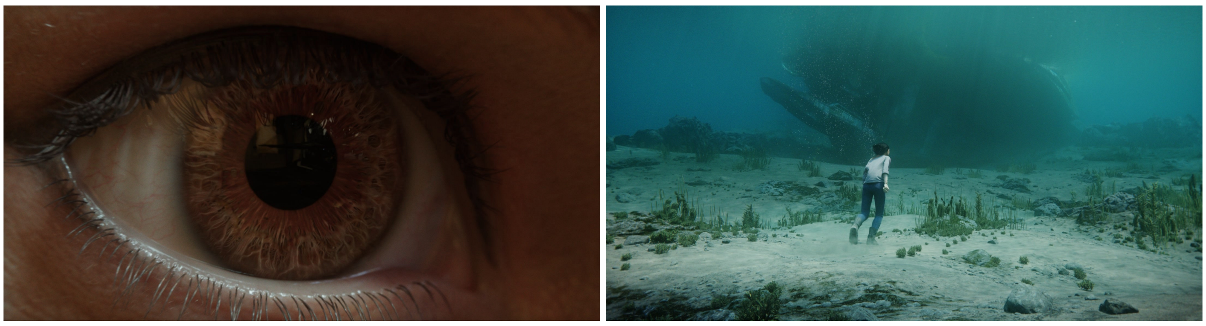


Fig. 1: Path guiding allows efficient rendering of notoriously difficult light transport conditions. These production shots from *Alita: Battle Angel* show its versatility. We were able to apply it on close-up shots of Alita's eyes featuring specular-diffuse-specular caustics as well as on vast underwater scenes with god-rays and caustics on the lake bed, in the volume, and on the main character. ©2018 Twentieth Century Fox Film Corporation. All rights reserved.

SIGGRAPH '19 Courses, July 28 - August 01, 2019, Los Angeles, CA, USA
ACM 978-1-4503-6307-5/19/07.
10.1145/3305366.3328091

Abstract

Path guiding is a family of adaptive variance reduction techniques in physically-based rendering, which includes methods for sampling both direct and indirect illumination, surfaces and volumes but also for sampling optimal path lengths and making splitting decisions. Since adoption of path tracing as a de facto standard in the VFX industry several years ago, there has been an increased interest in producing high-quality images with low amount of Monte Carlo samples per pixel. Path guiding, which has received attention in the research community in the past few years, has proven to be useful for this task and therefore has been adopted by Weta Digital. Recently, it has also been implemented in the Walt Disney Animation Studios' Hyperion and Pixar's Renderman. The goal of this course is to share our practical experience with path guiding in production and to provide self-contained overview of recently published techniques and to discuss their pros and cons. We also take audience through theoretical background of various path guiding methods which are mostly based on machine learning – used to adapt sampling distributons based on observed samples – and zero-variance random walk theory – used as a framework for combining different sampling decisions in an optimal way. At the end of our course we discuss open problems and invite researchers to further develop path guiding in their future work.

[1]The presented work was conducted while the author was employed at ETH Zürich and Disney Research.

Note

This manuscript is a draft. You can find the final version at
http://cgg.mff.cuni.cz/~jirka/path-guiding-in-production/2019/index.html

Contents

1 Objectives

In this course we share our practical experience that we gained from using guiding methods in Manuka, the production renderer at Weta Digital and from implementing guiding to Hyperion, the renderer used at Walt Disney Animation Studios. Another goal of this course is to bring order to recently published works on various guiding methods. Some are competing, some are orthogonal and some could be partially combined to form even better sampling schemes. We identify these relationships and we provide suggestions for important avenues of future research. We also cover introduction for people who are not familiar with guiding methods, covering theory and some concepts from various fields like machine learning or neutron transport that are behind guiding methods.

In summary, our main objectives are

- to share practical experience with guiding methods in production,
- to provide an overview of current guiding methods and discuss their strengths and weaknesses,
- to cover theoretical background to provide audience with solid understanding of path guiding, and
- to share open problems and possible research topics in the area with our community.

2 Syllabus

2.1 14:00 — Opening statement and introduction, Jiri Vorba (15 min)

We introduce path guiding as a family of methods for variance reduction in Monte Carlo based path tracers and state objectives of this course (see above). We provide motivation for adopting path guiding in production, give an overview of some typical scenarios where path guiding methods can greatly reduce numbers of traced paths (e.g. indirect illumination, caustics, selection between many lights) and we provide taxonomy of these methods based on the random decisions that they strive to improve.

2.2 14:15 — Theoretical Background, Jaroslav Krivanek (15 min)

View briefly review basic Monte Carlo rendering algorithms such as path tracing, light tracing, photon mapping or bidirectional path tracing. We define the meaning of "path guiding" as an attempt to perform globally optimal decisions in path construction. We discuss various data-driven approaches for learning the importance and relate it to basic problems in statistical learning such as density estimation and regression. We briefly discuss the relation to the theory of zero-variance random walk and we list the theoretical and practical requirements on good guiding methods such as computational efficiency, low memory overhead, minimal overhead in simple scenes, robustness etc.

2.3 14:15 — Bayesian Inference in Many-light Sampling, Jaroslav Krivanek (15 min)

Production scenes often feature tens, hundreds or even thousands of lights while only varying subsets are visible across the scene. This makes sampling of direct illumination challenging and unpredictable. If path guiding learns optimal sampling decisions only for indirect illumination, the variance coming from choosing a light source connection can still be inhibitive. Jaroslav will discuss a recent approach by Vévoda et al. [2018] to sampling many lights based on Bayesian inference. Bayesian learning is based on taking prior assumptions that are formed by subsequent observations (i.e. sampled paths) that provide evidence about real statistical properties of a given scene including occlusion.

2.4 14:45 — Guiding and Shadow Rays, Alexander Keller (30 min)

Alex will introduce an alternative data-driven approach to many-light sampling which computes and stores light source visibility first, and makes good use of this information during next-event estimation. This results in a guiding technique for shadow rays, and has the potential to greatly reduce noise in path tracing.

2.5 15:15 — Guiding methods in production, Thomas Müller (30 min)

We implemented the "Practical Path Guiding" algorithm [Müller et al., 2017] in Disney's Hyperion renderer [Burley et al., 2018] for use in movie production. Thomas will introduce the algorithm at a high level and describe three extensions based on recently published material Müller et al. [2018] that we developed to further improve the algorithm's effectiveness in a production environment. These extensions are (i) inverse-variance-weighted sample combination to avoid wasted samples, (ii) spatio-directional filtering to increase robustness against high-frequency illumination, and (iii) on-line learning of the BSDF : guiding ratio to improve handling of highly glossy materials. Mitsuba source code containing the extensions is available publicly at https://github.com/tom94/practical-path-guiding.

2.6 15:45 — Break (15min)

2.7 16:00 — Volumetric path guiding, Sebastian Herholz (30 min)

In participating media, path construction is influenced by scattering direction and distance sampling, Russian roulette, and splitting strategies. Sebastian will present a volumetric path construction technique Herholz et al. [2019] where all these sampling decisions are guided by a cached estimate of the adjoint light transport solution. This sampling strategy is based on the theory of zero-variance transport estimators, and it accounts for the spatial and directional variation in volumetric transport. Specifically, paths are constructed incrementally by sampling collision distances proportionally to the product of transmittance and the adjoint transport solution (i.e., in-scattered radiance). Scattering directions are likewise sampled according to the product of the phase function and the incident radiance estimate. Combined with an adaptive Russian roulette and splitting strategy tailored to volumes, variance is greatly reduced as compared to uni-directional methods. Sebastian will also discuss his experience with implementing this method in a production renderer such as Weta Digital's Manuka.

2.8 16:30 — Guiding in path space, Johannes Hanika (30 min)

Johannes will take the ideas presented in the last talk, especially about volume sampling and Russian roulette, and relate them to guiding of full paths in path space Simon et al. [2018]. Guiding new samples along full guide paths instead of marginalised distributions which only guide low dimensional parts at a time transparently includes all aspects. It also allows us to use simpler, uni-modal functions to represent a continuous PDF around guide samples. However both marginalised caches and full paths come with advantages and drawbacks. These properties will be systematically analysed and categorised, leading over seamlessly into the next talk.

2.9 17:00 — Open problems and future work, Jiri Vorba (15min)

We identify the most pressing problems and share them with the research community to enable further exploration of path guiding methods. We also discuss the possibility of combination between some of these methods so that the best of them would form more efficient algorithms.

Some of the aspects we cover are

- the problem of so called global exploration which is related to finding a useful path first before it can be locally explored,
- efficiency driven Russian roulette,
- problems specific to marginalised/cached guiding methods,
- problems related to "path space path guiding",
- problems specific to direct illumination sampling and light selection.

References

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Trans. on Graphics* 37, 3, Article 33 (Aug. 2018). https://doi.org/10.1145/3182159

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik Lensch, and Jaroslav Křivánek. 2019. Zero-Variance Based Sampling for Volume Path Guiding. *ACM Trans. on Graphics (conditionally accepted)* (2019).

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 36, 4 (June 2017), 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *CoRR* abs/1808.03856 (2018). arXiv:1808.03856 http://arxiv.org/abs/1808.03856

Florian Simon, Alisa Jung, Johannes Hanika, and Carsten Dachsbacher. 2018. Selective guided sampling with complete light transport paths. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Dec. 2018). https://doi.org/10.1145/3272127.3275030

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 37, 4 (Aug. 2018), 125:1–125:12.

3 Organizers

3.1 Jirí Vorba, Weta Digital

Jiří Vorba received his Ph.D. from *Charles University in Prague* in 2017. From September 2012 to January 2013, he undertook an internship at Max Planck Institute for Informatics in Saarbrücken under the supervision of Dr. Tobias Ritschel. In 2014, he implemented research results on path guiding achieved during his PhD. into *Manuka, Weta Digital's* renderer as part of his internship with *Weta Digital* in Wellington. Since 2017, he has been working as a researcher and *Manuka* developer for *Weta Digital*.

3.2 Johannes Hanika, Weta Digital

Johannes Hanika received his PhD in media informatics from *Ulm University* in 2011. After that he worked as a researcher for *Weta Digital* in Wellington, New Zealand. There he was co-architect of *Manuka*, *Weta Digital*'s physically-based spectral renderer. Since 2013 he is located in Germany and works as a post-doctoral fellow at the *Karlsruhe Institute of Technology* with emphasis on light transport simulation, continuing research for *Weta Digital* part-time. In 2009, Johannes founded the *darktable* open source project, a workflow tool for RAW photography. Johannes prefers limestone over sandstone for rock climbing.

4 Presenters

4.1 Sebastian Herholz, University of Tübingen

Sebastian Herholz is currently finishing his PhD in computer science at the University of Tübingen. His main research focus is on light transport algorithms, with a focus on importance sampling and guiding techniques for surface and volume rendering. From July to October 2017 he did an internship at the Charles University in Prague, where he worked together with Jaroslav Krivánek on algorithms for volumetric path guiding. During another internship at *Weta Digital* in Wellington, he was able to test the outcome of his research in a production environment. Sebastian is a passionate downhill mountain biker, who likes to travel around.

4.2 Thomas Müller, NVIDIA

Thomas Müller is a Research Scientist at NVIDIA, working on path-traced light-transport simulation and machine learning. He is currently finishing his PhD at ETH Zürich, where he also obtained his MSc (2016) and BSc (2014) degrees. Thomas' past research was conducted in collaboration with Disney and led to several awards and patents. The resulting algorithms were implemented in multiple production renderers, including Pixar's RenderMan and Walt Disney Animation's Hyperion. Prior to research, Thomas created commercial anti-cheat software for the MMORPG "FlyFF" and developed several core components of the online rhythm game "osu!".

4.3 Jaroslav Křivánek, Charles University/Render Legion

Jaroslav is an associate professor of computer science at Charles University, Prague, and a co-founder and the Direr of Research of Render Legion, the company behind Corona Renderer. Previously, he was affiliated with Cornell University, Czech Technical University, and University of Central Florida. Jaroslav received his Ph.D. from INRIA Rennes. His primary research interest is in realistic rendering, especially Monte Carlo methods for light transport simulation. His technology was adopted, among other, by Weta Digital, PIXAR, Allegorithmic, Chaos Group, etc. Jaroslav was selected for the New Europe 100 "list of outstanding challengers from Central and Eastern Europe".

4.4 Alexander Keller, NVIDIA

Alexander Keller is a Director of Research at NVIDIA. Before, he had been the Chief Scientist of mental images, where he had been responsible for research and the conception of future products and strategies including the design of the NVIDIA Iray light transport simulation and rendering system. Prior to industry, he worked as a full professor at Ulm University, where he co-founded the UZWR (Ulmer Zentrum für wissenschaftliches Rechnen) and received an award for excellence in teaching. Alexander Keller has more than 3 decades of experience in ray tracing and pioneered quasi-Monte Carlo methods for light transport simulation.

5 Introduction

Jiří Vorba, *Weta Digital Ltd.*

Since the advent of path tracing in movie production several years ago, the complexity of rendered scenes has grown substantially. They often include many hard-to-sample light transport effects like for example strong indirect illumination, occlusions, many lights, broad range of materials with various scattering profiles (BSDFs and phase functions), caustics, sub-surface scattering, hair and fur, and volumetric effects (Fig. 1). This challenges path tracing based on Monte Carlo integration which is infamous for its poor convergence rate. In practice, we need to trace many paths, average their contributions across each pixel and wait long tens of hours to obtain clean, noiseless image.

Thus production rendering systems usually apply several standard techniques to decrease the amount of noise. An essential one is adaptive sampling in the image plane which changes density of samples across pixels in order to equalize the noise amount. The idea is that light transport in some pixels is easier than in others and thus we can redistribute the sampling budget to donate more samples to problematic pixels. Another standard way to reduce the number of sampled paths is to employ denoising. Unfortunately, variance of samples in complex scenes is often so high that denoising still needs a relatively high number of samples to produce images without visible artifacts. The last resort taken in virtually all production systems is clamping of difficult light transport which reduces the photorealistic quality of rendered images.

These solutions do not address the primary source of the noise which comes from MC light transport simulation within the scene. To reduce the noise and thus rendering times, researchers have proposed many importance sampling schemes and their combinations through multiple importance sampling. The idea is that we invest some time into producing high quality samples. In other words most paths would ideally contribute significantly to the computed pixel while the number of paths that would be terminated without making any contribution would be reduced to minimum. Unfortunately, traditionally used sampling schemes are often imperfect mostly because they are local. That is they are not aware of global scene properties like configuration of geometry, materials and light sources within the scene. They thus *cannot adapt* to the given scene which results in repeating the same sampling patterns until convergence. For example even sophisticated algorithms based on multiple importance sampling of complete paths like *bidirectional path tracing* or *vertex connection and merging* often fail in scenes with non-trivial occlusion.

Thus recently, significant attention has been paid to *path guiding*, data-driven *adaptive* sampling schemes which can learn from sampled paths and can significantly improve MC convergence rate. The basic idea is that we can learn from previously sampled paths and harvest precious information about the light transport between light sources and camera. We can use this information to improve distribution of subsequently sampled paths in the scene so that they bring significant contributions to the sampled pixels.

5.1 Goals

In this course we would like to share our practical experience that we gained from implementing and using these methods in production renderers. Namely, we have been using guiding methods at Weta Digital in the Manuka renderer and recently, we implemented guiding to the Hyperion renderer used at Walt Disney Animation Studios. Further, we will share experience gained from our research in this area. One goal of this course is to sort and classify the many recently published works on various guiding methods [Keller and Dahm, 2019, Dahm and Keller, 2018, Herholz et al., 2016, 2019, Müller et al., 2017, Vorba et al., 2014, Vorba and Křivánek, 2016, Simon et al., 2018, Vévoda et al., 2018]. Some are competing, some are orthogonal and some could be partially combined to form even better sampling schemes. We will identify these relationships and we will provide suggestions for important avenues of future research in guiding methods, in the hope to spur a fruitful research community across the industry, to help shape the best possible light transport algorithm. We also provide an introduction for people who are not familiar with guiding methods, covering theory and some concepts from related fields like machine learning or neutron transport that are behind the adaptive sampling schemes.

5.2 Overview of the Course

say how we structure the course, what reader can expect from the following sections, make fast-forward for the reader

References

Ken Dahm and Alexander Keller. 2018. Learning Light Transport the Reinforced Way. In *Monte Carlo and Quasi-Monte Carlo Methods. MCQMC 2016. Proceedings in Mathematics & Statistics*, A. Owen and P. Glynn (Eds.), Vol. 241. Springer, 181–195.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 35, 4 (2016), 67–77.

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik Lensch, and Jaroslav Křivánek. 2019. Zero-Variance Based Sampling for Volume Path Guiding. *ACM Trans. on Graphics (conditionally accepted)* (2019).

Alexander Keller and Ken Dahm. 2019. Integral Equations and Machine Learning. *Mathematics and Computers in Simulation* abs/1712.06115 (2019).

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 36, 4 (June 2017), 91–100.

Florian Simon, Alisa Jung, Johannes Hanika, and Carsten Dachsbacher. 2018. Selective guided sampling with complete light transport paths. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Dec. 2018). https://doi.org/10.1145/3272127.3275030

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 37, 4 (Aug. 2018), 125:1–125:12.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 33, 4 (Aug. 2014), 101:1–101:11.

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 35, 4 (July 2016), 1–11.

6 Theoretical Background

Jaroslav Křivánek, *Charles University, Prague/Render Legion*

Note

This section is draft. You can find the final version at
http://cgg.mff.cuni.cz/~jirka/path-guiding-in-production/2019/index.html

6.1 Monte Carlo Light Transport Algorithms

View briefly review basic Monte Carlo rendering algorithms such as path tracing, light tracing, photon mapping or bidirectional path tracing.

6.2 Data-driven, Statistical Learning

We discuss various data-driven approaches for learning the importance and relate it to basic problems in statistical learning such as density estimation and regression.

6.3 Zero-variance Theory

We briefly discuss the relation of path guiding to the theory of zero-variance random walk.

6.4 Machine Learning Foundations

Subsequently, we discuss the basic machine learning concepts of the various existing methods such as maximum a posterior estimation, Bayesian inference, or online Expectation-Maximization and we will relate the basic concepts of the various existing methods to the requirements on successful guiding methods.

Algorithm 1: Off-line stepwise EM

```
// Index of sufficient statistics
$i := 0$
repeat
    // Iterate over a batch of $N$ samples
    for $q := 0$ to $N - 1$ do
        // E-step: Eq. (4)
        $\mathbf{u}_i^j :=$ UPDATESUFFICIENTSTATS($\mathbf{s}_q, \theta^{\mathrm{old}}$)
        if $i + 1 \bmod m = 0$ then
            // M-step: every $m$-th observed sample; Eq. (5)
            $\theta^{\mathrm{new}} := \overline{\theta}(\mathbf{u}_i^1, \ldots, \mathbf{u}_i^K)$
        end
        $i := i + 1$
    end
until CONVERGED();
```

7 Path Guiding

Jiří Vorba, *Weta Digital Ltd.*

By *path guiding* we refer to techniques which use a global knowledge about the scene to distribute transport paths. This knowledge, more specifically approximation of radiance field in the scene and additional sampling statistics, is learned from previous samples or in a pre-process step. As we incrementally construct paths vertex by vertex as described in Sec. 6.1, we *bias* random decisions taken in the process to *guide* the paths towards important regions in the scene. In these regions, paths are very likely to make significant contributions to the image. We can *bias* (i.e. change probability of) multiple decisions along each path, like choosing direction after each scattering event, free path sampling in volumes, absorption or choosing a light source for connection. Note, that biasing in this sense does not introduce bias (systematic error) in the image but results in the expected (correct) solution.

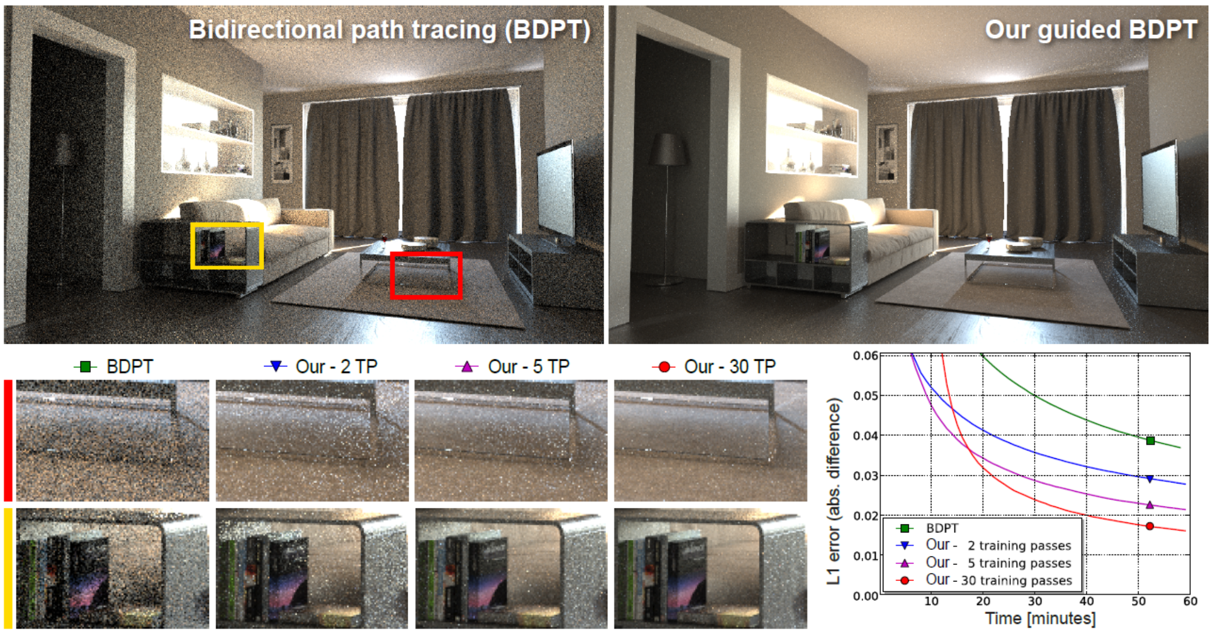


Figure 2: We illustrate the benefits of path guiding which learns from previous samples on a scene rendered by bi-directional path tracing (BDPT). Light transport in this scene is difficult for sampling because sun light enters the room through a small gap. Path guiding significantly reduces noise (left) as opposed to traditional BDPT. The path guiding performance depends on the number of samples used for learning the global information about the transport in the scene as shown in the insets and plots. The illustration is borrowed from work of [Vorba et al., 2014].

7.1 Applications

We can classify path guiding techniques according to their application. Path guiding can aid *path tracing* where path sampling starts at the camera as well as *light tracing* where sampling starts at light sources.

We begin by considering path tracing case as this is the most implemented algorithm in production. Path tracing in production scenes must efficiently handle sampling many lights, occlusion, indirect illumination, glossy materials, and ideally deal with caustic lighting. Various path guiding methods were developed over time to aid all these sampling problems. In next sections, we give an overview of path guiding methods for efficient sampling of both *direct illumination* (Sec. 7.2) and *indirect illumination* (Sec. 7.3). We pay special attention to *caustics* in Sec. 7.5, a special indirect illumination type, which are nearly impossible to render by path tracer without *directional guiding*. In Sec. 7.6, we briefly discuss performance of guiding methods on *glossy materials*.

Path guiding can learn to sample optimal path lengths and even suggest when we should split the traced paths to achieve superior performance, which is important for efficient sampling of indirect illumination. For this purpose, we describe *guided Russian roulette and splitting* in Sec. 7.7.

In the next section, Sec. 7.8, we discuss benefits of *guided emission* for light tracing and photons which enables using these algorithms in scenes of production scale. We continue by discussion on application of *path guiding* to bi-directional algorithms in Sec. 7.9.

We conclude by Sec. 7.10 and Sec. 7.11, where we discuss differences of various guiding solutions and list their desired properties respectively.

7.2 Direct illumination.

Computing direct illumination efficiently at every point in the scene is crucial for light transport performance. By sampling of direct illumination, we mean choosing the last segment of the path that is connected to a light source. It is important to realize that inefficient computation of direct illumination will also project into indirect illumination.

In most rendering systems, we have two sampling strategies for computing direct illumination. We either rely on hitting the light source with finite area or finite solid angle by *unidirectional* sampling (a reflected ray intersects the light source) or we explicitly sample a position/direction at the light source – a strategy known as *next-event estimation* (NEE). These two strategies are combined by multiple importance sampling (MIS) which weighs each sample contribution.

We understand path guiding as an adaptive method for sampling in the space of paths and as such, it can learn to improve both unidirectional and NEE sampling. In fact, there are various factors that can be learned from previous samples.

For unidirectional sampling, methods are the same as for indirect illumination. We learn directional distributions of direct illumination (or distributions that combine both direct and indirect illumination) and we cache them within the scene. We use them for sampling reflected rays which, in turn, increases the chance that we hit the light source. The benefit of these directional distributions is that they are aware of occlusion in the scene. However, this strategy is usually inferior to next-event estimation provided there is only one light source in the scene and no occlusion between illuminated surface and the light source.

Yet in production, we have to deal with scenes which often contain many occluded light sources. This pose a challenge for next-event estimation. To deal with large number of light sources with various geometric configuration and orientation, it is common to use some sort of hierarchical light source clustering Estevez and Kulla [2018], Fascione et al. [2019] and select a cut with candidate lights from which one light is sampled. However, these approaches are not aware of occlusion which can be learned per scene region to *guide the light selection*. We present two possible occlusion aware solutions to this many-lit problem in Sec. 8 and Sec. 9. The former one is implemented in Corona, a production renderer oriented on architecture visualization while the latter one was explored also in the real-time rendering context and can achieve interactive frame rates.

Yet another issue connected with NEE is the question *how many samples* we should take per one end vertex where we estimate the direct lighting. If the estimate is noisy, we would like to increase the number of samples to amortize the cost of tracing the whole path up to the position of the estimate. On the other hand, taking the NEE sample can be relatively costly as it usually includes descend through the hierarchy of lights. So in regions where the direct illumination is dim (or completely occluded), we would like to decrease the number of samples to bare minimum to save time. This number can even drop below 1 yet stay above 0 so that we only take NEE sample occasionally. In the same spirit, we can even learn how to distribute samples between distant lights and area lights.

Statistics needed for optimizing the NEE sample rates can be stored in caches within the scene (partitioning of the scene) which are used for guiding of indirect illumination. Thus, with majority of guiding solutions, it is possible to amortize the cost associated with search in the cache for the most relevant guiding record. The selected record will have the necessary statistics for sampling both direct and indirect illumination.

7.3 Indirect illumination.

We now consider illumination due to light scattering at least twice before reaching the camera. While such illumination appears smooth in the scene, it can be very difficult for path tracer to sample it efficiently. In essence, this happens due to (a) inefficient directional sampling and (b) sub-optimal path-length sampling. Both of these culprits can be mitigated by path guiding.

To explain inefficient directional sampling, we will consider a scene with lights close to geometry causing strong reflections on rather diffuse walls. Because these reflections occupy rather small solid angle when observed from distance, traditional sampling only occasionally finds the directions where the strong light is coming from. The reason is that such sampling is based on the surface properties and thus, for example on diffuse surfaces, spreads samples into a wide solid angle.

There are multiple different path guiding techniques available for *directional guiding* which learn optimal directions for sampling the rays in angular (directional) domain. These include early works on this topic which based foundation for future exploration Jensen [1995], Lafortune and Willems [1995], followed by works of Hey and Purgathofer [2002] and Bashford-Rogers et al. [2012]. Recently, more advanced techniques were introduced by Herholz et al. [2016, 2019], Müller et al. [2017], Vorba et al. [2014], Simon et al. [2018] that can handle mix of low and high frequency illumination and can be made practical for production. We discuss the details and differences between these advanced methods in Sec. 7.10, 10, 11, 12. For completeness, Dahm and Keller [2018] revealed interesting relationship between guiding and reinforcement learning, although they used the same representation as Jensen [1995] which does not scale well for complex production scenes. Also Müller et al. [2018] have shown that directional guiding is even possible using neural networks. However, network training relies on using GPUs and yet, the method approaches diminishing returns. We discuss the future potential of this method in Sec. 13.

Path sampling can benefit from directional guiding as long as we can efficiently decide whether it is worth to continue tracing the path or we should rather terminate it and start tracing a new one from the camera. It is important to guide this decision as it is a key to achieve high performance in simple scenes where most of the energy is transported over short paths as well as in more complex scenes where light undergoes many scattering events before reaching the camera. This problem is addressed by *guided Russian roulette and splitting* Vorba and Křivánek [2016] (also known as adjoint-driven Russian roulette and splitting) that we describe in Sec. 7.7.

7.4 Combination of Direct and Indirect Illumination

When we guide unidirectional path tracing in systems with next-event estimation, it is a question what quantity we should store in our directional distributions. It seems that the answer depends on how well our next-event estimation can importance sample the direct illumination.

If the next-event estimation is almost perfect and can handle occlusion as well as many-lights in the scene, the best choice is to simply ignore direct illumination completely and adapt only to indirect illumination. Otherwise, if we include direct illumination, which can be handled well in this example, we distract guiding from focusing on indirect illumination features since most guiding methods allocate limited resources (guiding resolution) on angles with high illumination. Thus, in turn, we would introduce variance into indirect illumination. Even worse, we would also introduce variance into direct illumination because unidirectional guiding for indirect illumination would be inferior strategy, however, good enough so that it is not completely down-weighted by MIS[2]

With guiding improvements to next-event estimation discussed in Sec. 7.2, it may sound that described methods almost form the perfect next-event estimation strategy. However, the practical experience says, that we always find a corner case in production scenes now and then which might be mitigated if the alternative strategy for sampling direct illumination could step in. Moreover, some shots might be illuminated by volumetric lights and large emissive particle systems which may be challenging for handling in light hierarchies and may require more specialized methods.

[2]Multiple importance sampling with balance or power heuristics is known for being sub-optimal. If a perfect strategy is combined with a moderate one, the noise of the estimator is increased.

We can see two ways for solving the dilemma whether to include/exclude direct illumination. First, we can keep separate distributions for each type (one for direct illumination and one for indirect) in the same spatial region in the scene. This would leave us with two strategies for unidirectional sampling and one NEE strategy. We might be able to learn the ideal mixing coefficients of all these strategies similarly to technique described in Sec. 10.5.

Simple solution to this problem is including direct illumination in the distribution but weighted by MIS weight. If next-event estimation is good strategy for a given direction then unidirectional samples will be down-weighted and unidirectional guiding can focus on indirect illumination or important directions that are not sampled well by NEE.

7.5 Caustics.

Caustics are type of indirect illumination that can benefit from *directional path guiding*. By its nature, caustics is light cast from a small light source and focused through refraction or by reflection of curved specular object. Caustic light is notoriously hard for path tracer to sample because light can come through extremely small solid angles. Quite often, in the production rendering systems, caustic light transport must be clamped to avoid unreasonable rendering times (Fig. 7.5).

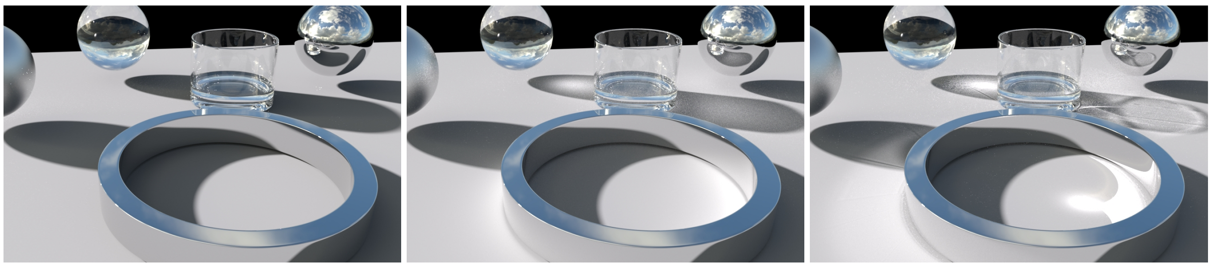


Figure 3: Scenes featuring caustics can look unrealistic when this kind of light transport is completely omitted (left) which is the default behavior in many path tracers. In fact, missing caustics around specular objects can trigger similar feeling as if the regular objects were missing shadows. Some production path tracers use strong filtering of caustic transport to retain some amount of lost energy (middle). Yet, the achieved look is still far from fully path traced caustics (right).

However, these interesting effects can add another level of realism (see Fig. 7.5) and thus production systems usually offer special solutions. For example, eyes of characters (Fig. 1) feature caustics visible through a refractive surface (so called specular-diffuse-specular transport). This specific case (eyes, water droplets) can be handled well by manifold exploration Hanika et al. [2015] although this approach can run into problems on geometry with highly varying curvature or in the presence of occlusion. Another, more complex option how to render the specular-diffuse-specular transport are photons Georgiev et al. [2012] (see Sec. 7.9). However, photon mapping needs a high density of photons sampled within the eyes which is not easy to achieve in scenes of production scale where eyes usually occupy only a tiny fraction of the scene.

On the other hand, *directional path guiding* allows to keep the simplicity of path tracer without running into the pitfalls like, for example, light leaks associated with photons, yet it is capable of rendering specular-diffuse-specular paths. Still, there are some limitations. A crucial assumption for path guiding is that light sources have a reasonable size and thus can be hit by unidirectional path tracing. Also directional path guiding uses approximations to model angular distributions of light so some amount of energy can turn into moderate fireflies. However, in contrast to using regular path tracing, these fireflies account for a very small fraction of total energy and can therefore be clamped in practice (Fig. 7.5b,c). We discuss more on rendering caustics in Sec. 7.9.

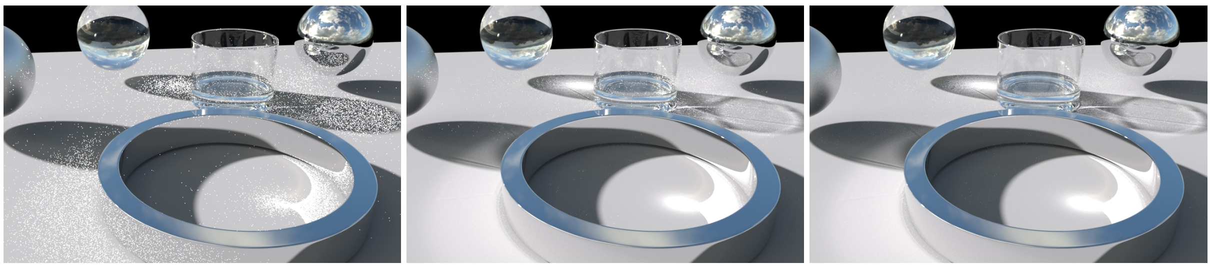


Figure 4: While rendering caustics by regular path tracing is possible in theory, the cost is unacceptable in practice (left). A convenient solution is extending path tracer by directional path guiding (middle) which is a general way to aid sampling of the whole range of indirect illumination effects. One problem of path guiding on this extremely difficult transport is that it generates fireflies. However, these fireflies capture only low amount of energy and can be clamped in practice without having a substantial impact on the look (right), leaving a uniform noise that is suitable for de-noising. All images were rendered with 1024 samples per pixel.

7.6 Glossy indirect illumination.

We can classify methods for directional guiding according to their capability of handling surfaces with low roughness or volumes with high mean cosine. Ideally, to generate high quality samples, choosing direction at a scattering event should consider product of incident illumination and BSDF (or phase function in volumes). Some guiding solutions Herholz et al. [2016, 2019] based on parametric mixtures are designed to handle glossy and glossy-to-glossy transport (see Sec.11) explicitly. In fact, the method by Herholz et al. [2016] is direct extension of the method by Vorba et al. [2014] to improve handling of the glossy transport. Recently proposed solution Simon et al. [2018] based on keeping full previously sampled paths inherently adjust to sampling the product distribution (see Sec. 12).

There are multiple options for solutions Müller et al. [2017], Vorba et al. [2014] that do not sample the product distribution but rather guide based on incident illumination – which is optimal for diffuse surfaces. Simplest solution is finding a fixed, global, threshold on roughness below which guiding the samples according to incident illumination is almost always inferior since response of the material or phase function would always yield almost zero contribution except for a small set of directions. On this materials we just resort to sampling the material of phase function.

Another option is using sampling-importance-resampling (SIR) Fan [2006] although distributing samples may take more time as it requires sampling of several candidates first and then a resampling step. More pressing issue associated with this method is the fact that we cannot evaluate exact probability of sampling a given direction which is problematic for MIS combination of unidirectional contributions and next-event estimation.

Last option is learning two sets of distributions in the scene. One set of distributions in the camera frustum that would learn the product distribution from samples with evaluated BSDF/phase function and would be used for guiding scattering of primary rays. The other set of distributions would be used for guiding subsequent bounces same as in the original methods.

Since the method from Müller et al. [2017] represents guiding distributions as quad trees it might be possible to consider using product trees. For more details refer to Sec. 10.

7.7 Guided Russian Roulette and Splitting

Path sampling can benefit from directional guiding as long as we can efficiently decide whether it is worth to continue tracing the path or we should rather terminate it and start tracing a new one from the camera. Traditionally, this decision called Russian roulette has been driven by albedo of surfaces or volumes.

However, scattering light many times in the scene before reaching the camera can result in premature termination of paths that significantly contribute to the image. As a result, their contribution turns into strong noise. This issue arises also in scenes with fully path-traced sub-surface scattering when path

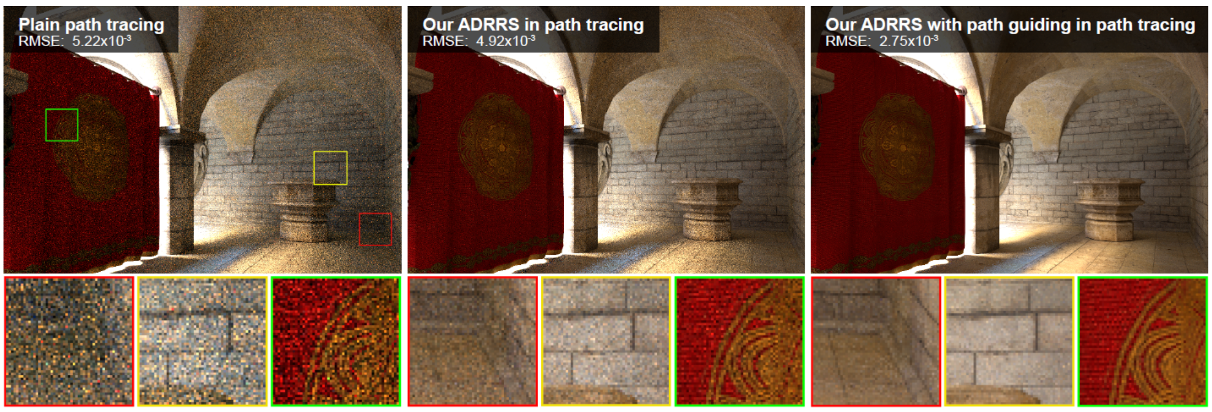

Plain path tracing
RMSE: 5.22x10⁻³
Our ADRRS in path tracing
RMSE: 4.92x10⁻³
Our ADRRS with path guiding in path tracing
RMSE: 2.75x10⁻³

Figure 5: In scenes where light is scattered many times before reaching the camera, good importance sampling and thus noise reduction can be achieved by guided (adjoint-driven) Russian roulette and splitting (ADRRS). Using traditional albedo-driven Russian roulette in path tracing is sub-optimal under these conditions (left) because paths are either terminated too soon or time is wasted on sampling overly long paths. Using global knowledge about the scene clearly reduces noise in indirectly lit regions (middle). Directional path guiding can be naturally combined with ADRRS which results in synergic noise reduction (right). Image courtesy of Vorba and Křivánek [2016].

emerges from the object and is terminated at the next vertex without being able to connect to a light. On the other hand, albedo driven termination may result into spending too much time on tracing reflections between materials with high albedo (white walls, snow, blond hair, white fur, etc.).

To remedy this, Vorba and Křivánek [2016] introduced *guided Russian roulette and splitting* (also known as adjoint-driven Russian roulette and splitting) which allows to optimize path termination by using global knowledge about the scene learned from previous samples (see Fig. 7.7). Moreover, path guiding can use this knowledge to split the path in important regions which are, in turn, covered by more samples. Increased efficiency follows from the fact that path splitting amortizes the work spent on tracing the whole path up to the splitting point.

The more scattering events along the path the greater benefit the guided Russian roulette and splitting provides. This is a reason why it is so important for efficient volumetric transport where average path length is usually high. In Sec. 11, we describe this technique in detail within a full path guiding solution for volumetric and surface transport. An alternative guiding method which inherently combines Russian roulette and directional decisions is presented in Sec. 12

7.8 Guided Light/Photon Tracing Emission

Using light tracing or photons for rendering caustics in production scenes is often not possible due to the size of the scene with respect to its portion visible in the camera frustum. Scenes are usually large for various reasons, some of them being (a) scenes are modeled for multiple different shots or (b) camera is flying through the scene within one shot or (c) just for the case a director would decide to change the camera in the given shot in later production stages. Moreover, large open environments like for example lakes and oceans with distant horizons cannot even be made smaller because this would become visible in the image. Note, that this problem with large scenes projects even to bi-directional methods like bi-directional path tracing Lafortune and Willems [1993], Veach and Guibas [1994], vcm/ups Georgiev et al. [2012], Hachisuka et al. [2012], or upbp Křivánek et al. [2014] which, in turn, can degrade to rather expensive path tracers.

The reason why large scenes are prohibitive for photons is that chances of sampling corresponding important light paths is very small and thus, in turn, their density is low in important places within the camera frustum. We identify the sampling of light path emission from distant light sources as the major culprit.

The distant light sources, like sun and sky or HDR environment maps, are usually used to lit open environments but also interiors through openings and windows. The fundamental property of distant light sources is that emitted radiance from any direction is constant with respect to every point in the scene unless the point is in the shadow due to occlusion. Thus, when a light path is emitted, we first need to decide from what position the path should start and in what direction we should trace the first ray. Common solution is to first sample a direction as if we sampled distant lights for next-event estimation. Then the starting position is sampled uniformly within the area that spans the scene bounds and which is perpendicular to the sampled direction. Furthermore, the position must be outside of the scene bounds so that every un-occluded point in the scene has a chance to receive the illumination. However, this conservative strategy becomes sub-optimal quite fast with increasing extent of the scene.

Path Guiding. Instead, at Weta Digital, we use path guiding to learn the optimal sampling distribution for sampling the position of emitted light paths based on method described by Vorba et al. [2014]. This method is based on keeping a number of 2D guiding distributions for position sampling. Each distribution is relevant for a compact set of directions while the union of sets form the whole sphere of directions. In other words, after sampling the initial direction of the emitted light path, we find corresponding guiding distribution for sampling the starting position.

Each guiding distribution is proportional to the camera importance reflected from the scene within the corresponding set of directions. In this way, the density of photons near the camera will become high whereas in places further away or tilted with respect to the camera view-direction will have lower density. Note, that scene points that are not visible in the image and do not even reflect indirect illumination into the image do not receive any photons. For bi-directional methods, this could be combined with approach described recently by Grittmann et al. [2018] which includes MIS weights in the guiding distributions to detect where photons are not needed due to existence of other more efficient strategies.

For distant light sources, it makes sense to guide also the direction selection of the initial ray based on the product of camera importance and emitted radiance as described in the original method by Vorba et al. [2014]. For example, the scene can be illuminated only by part of the environment while its brightest part might be actually occluded (imagine interior illuminated through a window when sun is on the other side of the building).

Extensions. While Vorba et al. [2014] train the guiding distributions for light tracing emission in the pre-processing step from paths traced from the camera, we have found that it is possible to extend the *guided emission* so that the 2D distributions are trained on-line from forward samples similarly as Müller et al. [2017] do for path tracing (see Sec. 7.10 for the details about forward/reverse guiding). However, there is a problem associated with this approach.

Because we have no a-priory knowledge about the target guiding distribution, we use the classical uniform sampling for emitting the light paths in the early stages. As a result, first samples that contribute to the image have usually high variance and in many cases they just become strong fireflies. As guiding explores the domain based on these first initial non-zero contributions, it starts to provide reasonable low-variance contributions. Thus we have to deal with the first noisy contributions that can be very difficult to "average out" from the final image. Note, that for large scenes, the difficulty of sampling the emission position in light/photon tracer becomes equivalent to sampling caustics due to very small light source in the path tracer.

To address this issue we run a very short phase when we warm-up the guiding caches by emitting a batch of initial paths that are not splat into the beauty image. Further, we try to cull the area which needs to be explored by guiding. A conservative approach would be to take the area given by the scene (geometry) bounds (as described for the uniform sampling of emission positions at the beginning of this section). We instead, within the short pre-pass, trace also a batch of paths from the camera which gives us a point cloud that allows us to determine the actual scene extent. This helps guiding to find non-zero contributions faster.

Note, that this problem of noisy initial estimates could also be mitigated by weighting contributions by their inverse variance as described in Sec. 10.3.

7.9 Path Guiding and Advanced Bi-directional Algorithms

As discussed in Sec. 7.3 and 7.5, path guiding allows efficient rendering of directional dependent effects like caustics or indirect reflections of light sources that are close to geometry using only path tracer. These are examples of effects that otherwise require more complex bi-directional transport algorithms like vcm introduced by Georgiev et al. [2012] which combine paths traced from lights and paths from camera.

Bi-directional Drawbacks. Being able to stick to path tracer is suitable for production as path tracer is the most favored light transport algorithm as described recently by Burley et al. [2018], Christensen et al. [2018], Fascione et al. [2018], Georgiev et al. [2018], Kulla et al. [2018]. Not only because it is relatively simple for implementation but it can accommodate many tricks that are difficult or impossible to incorporate in bidirectional algorithms. For example, one can completely change shading for indirect illumination or increase roughness after multiple scattering events. Another example is so call *point-of-entry* sub-surface scattering when the volume properties are derived from a texture on the object boundary. These tricks usually depend on the the path prefix which makes it challenging to handle within strict bi-directional constraints. When tracing paths from light sources it is not clear what scattering probabilities one should use unless the path is connected to the camera. Also the path evaluation must be deferred until the full path is sampled.

Another pressing issue of bi-directional methods is that they usually double the amount of traced paths per one progression[3] and also spend some time on their combination. This is not a problem when the extra cost is amortized in difficult transport scenarios by significant decrease of noise in each progression which, in turn, results in faster render thanks to the decreased total number of progressions. However, the majority of production scenes feature such conditions that many contributions of bi-directional combinations are down-weighted (and thus discarded) Grittmann et al. [2018], meaning that path tracing is a better choice.

Path Guiding. The natural quesiton is "Can we aid bi-directional algorithms by path guiding to get faster renders and remedy some of their issues?" Yes, we can and indeed, as we discussed in Sec. 7.8, thanks to guided emission, it can deliver light paths in front of the camera so that combination of paths, which is the source of the bi-directional strength, is even possible. Moreover, we can guide all sampling decisions along each path (both paths from the camera and from light sources). Yet, as shown in the work by Vorba et al. [2014] it seems that guided path tracing is more or less as efficient as guided bi-directional algorithms in the challenging scenarios (Fig. 7.9), meaning that many paths are redundant and their contribution is strongly down-weighted by MIS. Because bi-directional algorithms are unnecessarily computationally heavy in most production scenes, we also favor path tracing as the default light transport algorithm.

However, we found at Weta Digital, that thanks to guiding, we can make the bi-directional algorithms relatively lightweight if we retain only path tracing with next-event estimation and light tracing, which connects to the camera. We do not interconnect neither merge paths. We use this for shots where caustic lighting is received by majority of pixels because light tracing with guided emission (see Sec. 7.8) can be still superior to guided path tracing (see the paragraph *Caustics* in Sec. 7.1). This is true especially for caustics coming from very tiny light sources.

7.10 Different Approaches

different technical implementations of the same idea, different distributions, some properties/limitations come from these technical choises -> product sampling; memory consumption; training speed/sampling

[3]By progression we mean a fixed amount of paths usually equal to the number of pixels for both path tracer and light tracer.

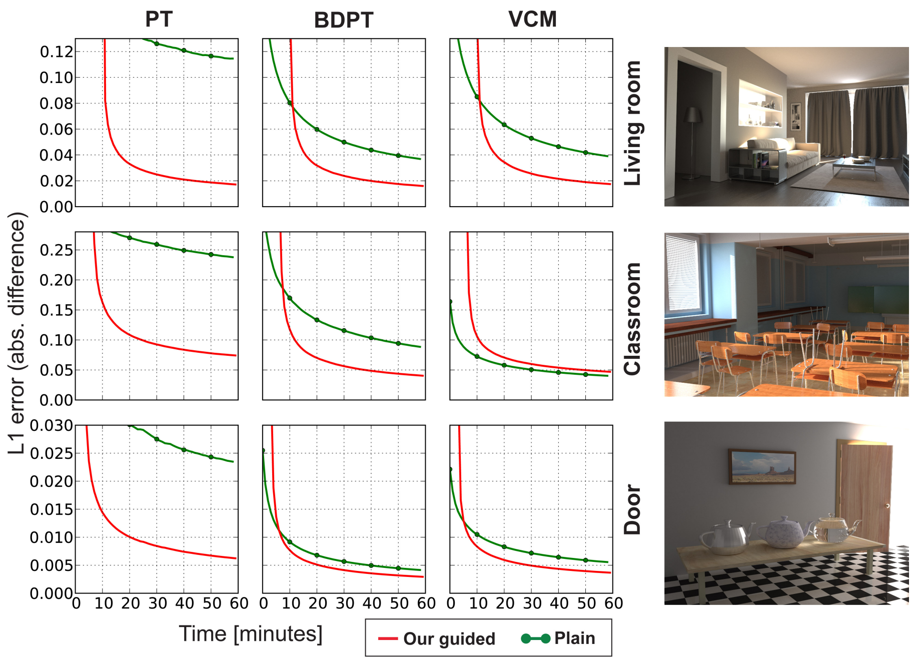


Figure 6: Time dependence of L_1 error for 60 minutes of rendering. These graphs show that guided path tracer performs almost as well as guided bi-directional methods (bi-directional path tracing – middle, vertex connection and merging – right) in three different scenes while the performance gap of unguided methods is substantial. We borrowed the figure from work of Vorba et al. [2014].

speed; pre-process/no-preprocess, spatial cache/no spatial cache,

7.11 Requirements on Successful Guiding Methods

We list the theoretical and practical requirements on good guiding methods such as computational efficiency, low memory overhead, minimal overhead in simple scenes, robustness etc.

References

Thomas Bashford-Rogers, Kurt Debattista, and Alan Chalmers. 2012. A Significance Cache for Accelerating Global Illumination. *Computer Graphics Forum* 31, 6 (2012), 1837–51.

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney&#x02019;s Hyperion Renderer. *ACM Trans. Graph.* 37, 3, Article 33 (July 2018), 22 pages. https://doi.org/10.1145/3182159

Per Christensen, Julian Fong, Jonathan Shade, Wayne Wooten, Brenden Schubert, Andrew Kensler, Stephen Friedman, Charlie Kilpatrick, Cliff Ramshaw, Marc Bannister, Brenton Rayner, Jonathan Brouillat, and Max Liani. 2018. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Trans. Graph.* 37, 3, Article 30 (Aug. 2018), 21 pages. https://doi.org/10.1145/3182162

Ken Dahm and Alexander Keller. 2018. Learning Light Transport the Reinforced Way. In *Monte Carlo and Quasi-Monte Carlo Methods. MCQMC 2016. Proceedings in Mathematics & Statistics*, A. Owen and P. Glynn (Eds.), Vol. 241. Springer, 181–195.

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 25 (Aug. 2018), 17 pages. https://doi.org/10.1145/3233305

Shaohua Fan. 2006. Sequential Monte Carlo Methods for Physically Based Rendering. (2006).

Luca Fascione, Johannes Hanika, Daniel Heckenberg, Christopher Kulla, Mark Droske, and Jorge Schwarzhaupt. 2019. Path Tracing in Production – Part1. In *SIGGRAPH Courses*.

Luca Fascione, Johannes Hanika, Mark Leone, Marc Droske, Jorge Schwarzhaupt, Tomáš Davidovič, Andrea Weidlich, and Johannes Meng. 2018. Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production. *ACM Trans. Graph.* 37, 3, Article 31 (Aug. 2018), 18 pages. https://doi.org/10.1145/3182161

Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* 37, 3, Article 32 (Aug. 2018), 12 pages. https://doi.org/10.1145/3182160

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Pascal Grittmann, Arsène Pérard-Gayot, Philipp Slusallek, and Jaroslav Křivánek. 2018. Efficient Caustic Rendering with Lightweight Photon Mapping. *Computer Graphics Forum* 37, 4 (2018). EGSR '18.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. Graph.* 31, 6, Article 191 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366210

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34, 4 (June 2015), 87–97.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 35, 4 (2016), 67–77.

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik Lensch, and Jaroslav Křivánek. 2019. Zero-Variance Based Sampling for Volume Path Guiding. *ACM Trans. on Graphics (conditionally accepted)* (2019).

Heinrich Hey and Werner Purgathofer. 2002. Importance sampling with hemi-spherical particle footprints. In *Proceedings of the 18th Spring Conference on Computer Graphics*. 107–114.

H. Jensen. 1995. Importance Driven Path Tracing Using the Photon Map. In *Rendering Techniques 1995 (Proc. 6th Eurographics Workshop on Rendering)*, P. Hanrahan and W. Purgathofer (Eds.). Springer, 326–335.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.* 33, 4 (Aug. 2014), 1–13.

Christopher Kulla, Alejandro Conty, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. *ACM Trans. Graph.* 37, 3, Article 29 (Aug. 2018), 18 pages. https://doi.org/10.1145/3180495

E. Lafortune and Y. Willems. 1995. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Rendering Techniques 1995 (Proc. 6th Eurographics Workshop on Rendering)*, P. Hanrahan and W. Purgathofer (Eds.). Springer, 11–20.

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *PROCEEDINGS OF THIRD INTERNATIONAL CONFERENCE ON COMPUTATIONAL GRAPHICS AND VISUALIZATION TECHNIQUES (COMPUGRAPHICS '93*. 145–153.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* 36, 4 (June 2017), 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *CoRR* abs/1808.03856 (2018). arXiv:1808.03856 http://arxiv.org/abs/1808.03856

Florian Simon, Alisa Jung, Johannes Hanika, and Carsten Dachsbacher. 2018. Selective guided sampling with complete light transport paths. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Dec. 2018). https://doi.org/10.1145/3272127.3275030

Eric Veach and Leonidas Guibas. 1994. Bidirectional estimators for light transport. In *In EGWR '94*. 147–162.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 33, 4 (Aug. 2014), 101:1–101:11.

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 35, 4 (July 2016), 1–11.

8 Bayesian Inference in Many-light Sampling

Jaroslav Křivánek, *Charles University, Prague/Render Legion*

Note

This section is draft. You can find the final version at
http://cgg.mff.cuni.cz/~jirka/path-guiding-in-production/2019/index.html

Complete guiding solution that can stand the load of production scenes needs to adapt both sampling of indirect and direct illumination. Production scenes often feature tens, hundreds or even thousands of lights while only varying subsets are visible across the scene. This makes sampling of direct illumination challenging and unpredictable. If path guiding learns optimal sampling decisions only for indirect illumination the variance coming from choosing a light source connection can still be inhibitive.

In this part we discuss a recent approach [Vévoda et al., 2018] to sampling many lights based on Bayesian inference. As explained in the previous section, Bayesian learning is based on taking prior assumptions that are formed by subsequent observations (i.e. sampled paths) that provide evidence about real statistical properties of a given scene. This Bayesian approach turns out to be advantageous for robust data-driven learning and applicable to both many-light sampling and indirect illumination path guiding [Vorba et al., 2014].

9 Guiding and Shadow Rays

Alexander Keller, *NVIDIA*

At the core of physically based light transport simulation are materials specified by bidirectional scattering distribution functions (BSDF). Given a direction of incidence and a second direction into which radiance is scattered, these functions determine the fraction of radiance transported for any point of the scene surface. Besides evaluation, determining a scattering direction given a surface point and a direction of incidence is the second basic operation that is implemented in modern rendering algorithms. Scattering allows for tracing light transport paths, for example creating photon trajectories starting at light sources.

Light transport simulation consists of sampling light transport paths and summing up their contribution to the pixels of an image. Using scattering, this is most efficient, if paths are sampled proportional to their probability, which includes to learn where light comes from [Dahm and Keller, 2018, Keller and Dahm, 2019]. Based on early ideas by Lafortune and Willems [1995] and Jensen [1995], guiding paths has made huge progress recently [Herholz et al., 2016, Müller et al., 2017, Müller et al., 2018, Zheng and Zwicker, 2018].

However, when connecting endpoints of segments of paths by shadow rays to test their mutual visibility, the evaluation of the BSDF does not consider visibility. Hence images may be noisier when paths are not sampled proportional to their contribution.

The course notes are based on own work and almost identical work done independently [Mikolajewski, 2018]. It deals with efficient approaches to improve next-event estimation by including information about visibility in order to make shadow rays much more efficient. After briefly reviewing the principle of importance sampling in Sec. 9.1, we introduce the concept of the partial cumulative distribution function in Sec. 9.2 and its stochastic interpolation in Sec. 9.2.2. Realizing that a hitpoint of a photon is the end of a ray that started where light came from, allows one to use photon trajectories to include an informed guess about visibility and thus to notably reduce noise in image synthesis, which is explored in Sec. 9.3. The probabilities for next-event estimation may be learned during guided path tracing, as introduced in Sec. 9.4.

9.1 Importance Sampling

Light transport simulation [Pharr et al., 2016] relies on Monte Carlo [Sobol', 1991] and quasi-Monte Carlo methods [Keller, 2013] for approximating integrals, as general closed form solutions are not available. One important method of improving the efficiency of estimating integrals by averages of samples of the integrand is importance sampling, which is based on a simple transformation:

$$\int_{[0,1)^s} f(x)dx = \int_{[0,1)^s} f(x)\frac{p(x)}{p(x)}dx \approx \frac{1}{N}\sum_{i=0}^{N-1}\frac{f(y_i)}{p(y_i)}$$

Importance sampling would be perfect, if the probability density function p was proportional to the integrand f, However, this would imply that we know the integral already. In practice, we therefore generate samples $\{y_0, \ldots, y_{N-1}\}$ with a distribution as proportional to p as possible.

As mentioned in the previous section, importance sampling by guiding light transport paths to where radiance comes from is quite advanced. However, besides selecting scattering directions, simulation efficiency may be improved by sampling points on light sources and shooting a shadow ray to check, whether they illuminate a point. Therefore the integral over the hemisphere $\mathcal{S}^2_-(x)$ aligned by the surface normal in x representing the reflected radiance L_r in a point x in direction ω is transformed to an integral over

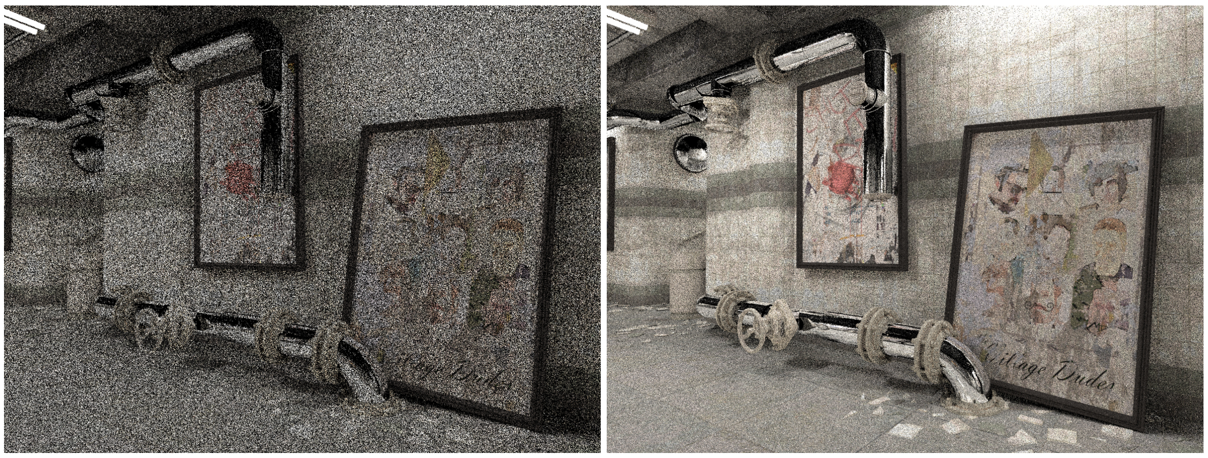


Figure 7: Comparison. Left: Uniform sampling. Right: The effect of resampling importance sampling: First, 8 samples are generated from a uniform distribution across the light sources. Then a cumulative distribution function (CDF) is determined using the unoccluded contribution of each of the initial samples. The actual shadow rays to be shot are determined by resampling the CDF. Unreal Engine demo by Epic Games.

the surface ∂V of the geometry:

$$\begin{aligned} L_r(x,\omega_r) &= \int_{S^2_-(x)} L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x d\omega \\ &= \int_{\partial V} V(x,y) L_i(x,\omega) f_r(\omega_r,x,\omega)\cos\theta_x \frac{\cos\theta_y}{|x-y|^2} dy \end{aligned} \tag{1}$$

The visibility function $V(x,y)$ is either one, if both points can see each other and zero if they are occluded. f_r represents the bidirectional scattering distribution function and L_i is the incident radiance.

The numerical evaluation of equation (1) is called next-event estimation and has been intensively investigated, see for example Vévoda and Křivánek [2016] and Vévoda et al. [2018]. The main challenge of importance sampling is the inclusion of the visibility function V. We therefore briefly review some fundamental approaches in the following sections before proposing our approach.

9.1.1 Resampling Importance Sampling

Resampling importance sampling is a general concept that can be used in combination with any importance sampling method. The idea is to initially sample from a first distribution, for example creating samples uniformly distributed across the surface of the area light sources. Then a cumulative distribution function (CDF) is determined given a second distribution, for example the unoccluded contribution of each previous light source sample. The actual shadow rays to be shot then are selected according to this CDF [Bikker, 2012, Talbot et al., 2005].

A comparison between uniform sampling and resampling importance sampling is shown in Figure 7. While a larger number of initial samples results in a more accurate importance sampling, it comes at the cost of more evaluations of the second distribution. This can be quite expensive, although no shadow rays are involved, yet. It is therefore crucial to carefully choose the number of initial samples for the resampling CDF.

9.1.2 Light Hierarchies

In a pre-processing step a binary tree is built over the emissive geometry in a scene. During rendering, this tree is probabilistically traversed depending on local properties like for example position and solid angle.

Reaching a leaf, a light source is selected either by uniformly sampling or by sampling a pre-computed cumulative distribution function (CDF).

While light hierarchies are standard in industry [Estevez and Kulla, 2018, 2017, Keller et al., 2017] and even perform in realtime [Moreau and Clarberg, 2019], they lack the inclusion of actual visibility information, which may dramatically decrease their efficiency in certain situations.

9.1.3 Including Visibility

Already Ward in his RADIANCE rendering system [Ward, 1991, 1994] used visibility information to reduce variance: For each light source the number of successful shadow rays and the total number of shadow rays were tracked and their ratio served as an approximation of average visibility.

Then Jensen and Christensen [1995] traced rays straight through objects to create shadow photons to indicate the lack of illumination. As the number of shadow photons to deposit depends on the depth complexity of the scene, the method became less practical with more complex geometry.

When querying photons in a photon map, the origins of the photons indicate locations where light may be coming from. This idea has been used by Keller and Wald [2000]: Shadow rays were traced to photon origins and for consistency purposes one ray has been traced to a randomly selected light source not represented by the queried photons. In a similar fashion, probabilistic connections for bidirectional path tracing [Popov et al., 2015] determine probability distribution functions by subsampling all possible connections between light and camera path segments. Then the probability density functions closest to the point to be shaded are interpolated and used to select the actual shadow rays to be shot.

9.2 Partial Cumulative Distribution Functions

In order to save memory, we take advantage of the fact that out of the many light sources that may influence a point to be shaded often only a small number has a probability that is vastly different from uniform. Then the probability of the i-th light source can be written as

$$p_i := \begin{cases} (1-b)\cdot q_i + b\cdot \frac{1}{n} & \text{for } i \in I \\ b \cdot \frac{1}{n} & \text{for } i \notin I, \end{cases} \tag{2}$$

where the index set $I := \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ is a subset of the set of all indices of light sources. Note that $\sum_{i=1}^{n} p_i = 1$.

A *partial cumulative distribution function (CDF)* $Q_k := \sum_{j=1}^{k} q_{i_j}$ now only stores the non-uniform part of the distribution. The uniform part is implicitly given by the constant $b \in [0, 1]$.

In order to sample a light source index, with probability $1 - b$ the partial CDF is used for selection, while with probability b any of the n light sources is selected uniformly. For a survey of efficient parallel algorithms to sample a CDF, see Binder and Keller [2019].

In the subsequent sections, we explain how to store partial CDFs in a scene, how to determine the index set I, and how to efficiently compute the probabilities q_i.

9.2.1 Multiple Importance Sampling

Instead of sampling the uniform part of a partial cumulative distribution function, for example, a light hierarchy can be used to improve results. We combine both methods using multiple importance sampling. Each sample is weighted by the inverse of the sum of the probability of creating it with the partial cumulative distribution function and the probability of creating it using the light hierarchy. Note that these weights replace the constant b in equation 2, which otherwise needs to be a fixed small value or may be learned.

Algorithm 1: Computation of the two hashes used for lookup. Note that the arguments of a hash function, which form the descriptor, may be extended to refine clustering, see Binder et al. [2019].

Input: Location x of the vertex, the normal n, the position of the camera p_{cam}, and the user-selected scale s.
Output: Hash i to determine the position in the hash table and hash v for verification.

```
1 $l \leftarrow \text{level\_of\_detail}(|p_{cam} - x|)$
2 $x' \leftarrow x + \text{jitter}(n) \cdot s \cdot 2^l$
3 $l' \leftarrow \text{level\_of\_detail}(|p_{cam} - x'|)$
4 $\tilde{x} \leftarrow \left\lfloor \frac{x'}{s \cdot 2^{l'}} \right\rfloor$
5 $i \leftarrow \text{hash}(\tilde{x}, \ldots)$
6 $v \leftarrow \text{hash2}(\tilde{x}, n, \ldots)$
```

9.2.2 Hashing and Stochastic Interpolation

In order to locally adapt the partial cumulative distribution functions, space may be partitioned into a uniform grid of voxels or by a *k*d-tree. Even more efficient, we may use the hashing scheme introduced by Binder et al. [2019]: We select a hash table size proportional to the number of pixels on the screen, where each hash table entry stores one partial cumulative distribution function. The code fragment in algorithm 1 illustrates how to look up a voxel given a location x in space. Jittering the lookup position replaces an explicit blending of distributions and can be considered stochastic interpolation. For more detail, we refer to Binder et al. [2019].

9.3 Photon-Guided Shadow Rays

For next-event estimation, we trace primary photons (i.e. light path segments of length 1 starting on light sources) and store these in a spatial data structure (also see Mikolajewski [2018]). For each voxel in this data structure a CDF is built over the contributions of the photons as seen from the camera. When computing the actual image, camera paths are traced and at each path vertex an explicit connection to a light source is established. To select a light source, the CDF of the voxel containing the vertex is sampled, which in turn means that a light source is selected proportional to its contribution including visibility. In addition we combined our method with a light hierarchy using multiple importance sampling to overcome the problem of low photon density regions. This way our combined solution takes the best of both worlds and never performs worse than the light hierarchy.

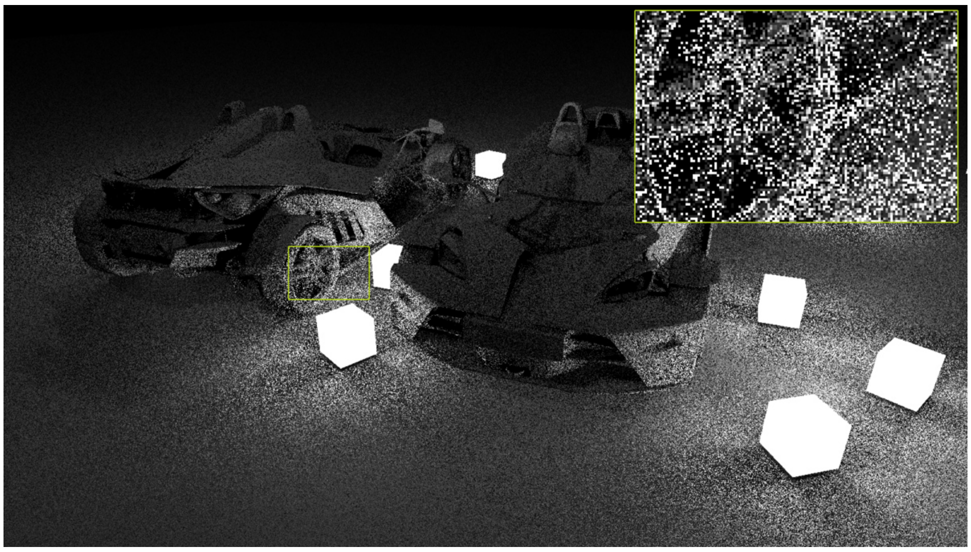
The benefits of the technique are shown in Fig. 8 and Fig. 9. The model shown in Fig. 10 contains 10624 area light sources and is rendered using 4 paths of length 4 per pixel. While a light hierarchy still suffers from high variance due to light selection without taking into account visibility, our method may reduce this source of variance. In addition our method is faster than probabilistically traversing a hierarchy. With a budget of 7x4 rays per pixel, our method runs at 24FPS in a pure software GPU implementation. Note that the remaining noise is very uniform in nature and hence very amenable to filtering: Fig. 10 includes a combination of the sampling technique with path space filtering [Binder et al., 2019, Keller et al., 2016].

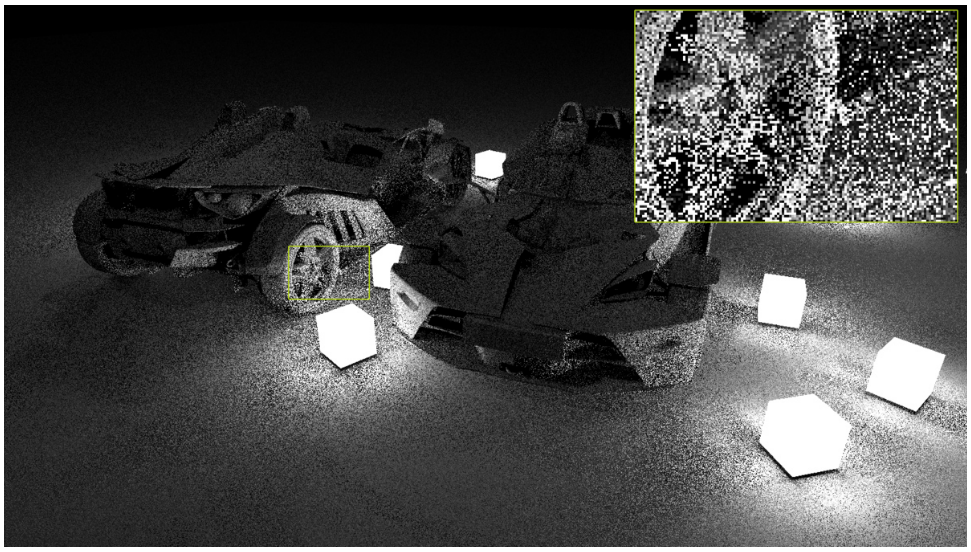
9.4 Learning which Shadow Rays to Shoot

Instead of using photons, tracing a small number of paths across the image before rendering is sufficient to estimate the average contributions of the light sources across the image [Wald et al., 2003]. Normalizing the contributions yields a probability density for sampling during rendering. Even neural networks can be used to learn probabilities for next-event estimation [Keller and Dahm, 2019].

9.5 Conclusion

Next-event estimation helps to increase the efficiency of light transport simulation, especially, if importance sampling includes information about the actual visibility. We proposed the concept of a hash table storing partial cumulative distribution functions. Accessing the data structure by stochastic interpolation


(a) Uniform sampling


(b) Light hierarchy

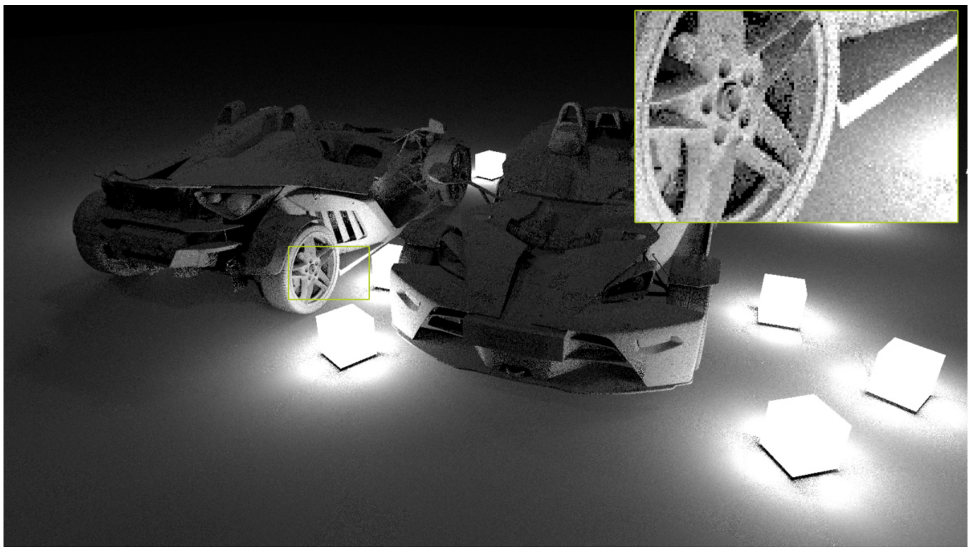

(c) Visibility-guided importance sampling

Figure 8: Method comparison at 16 paths per pixel.

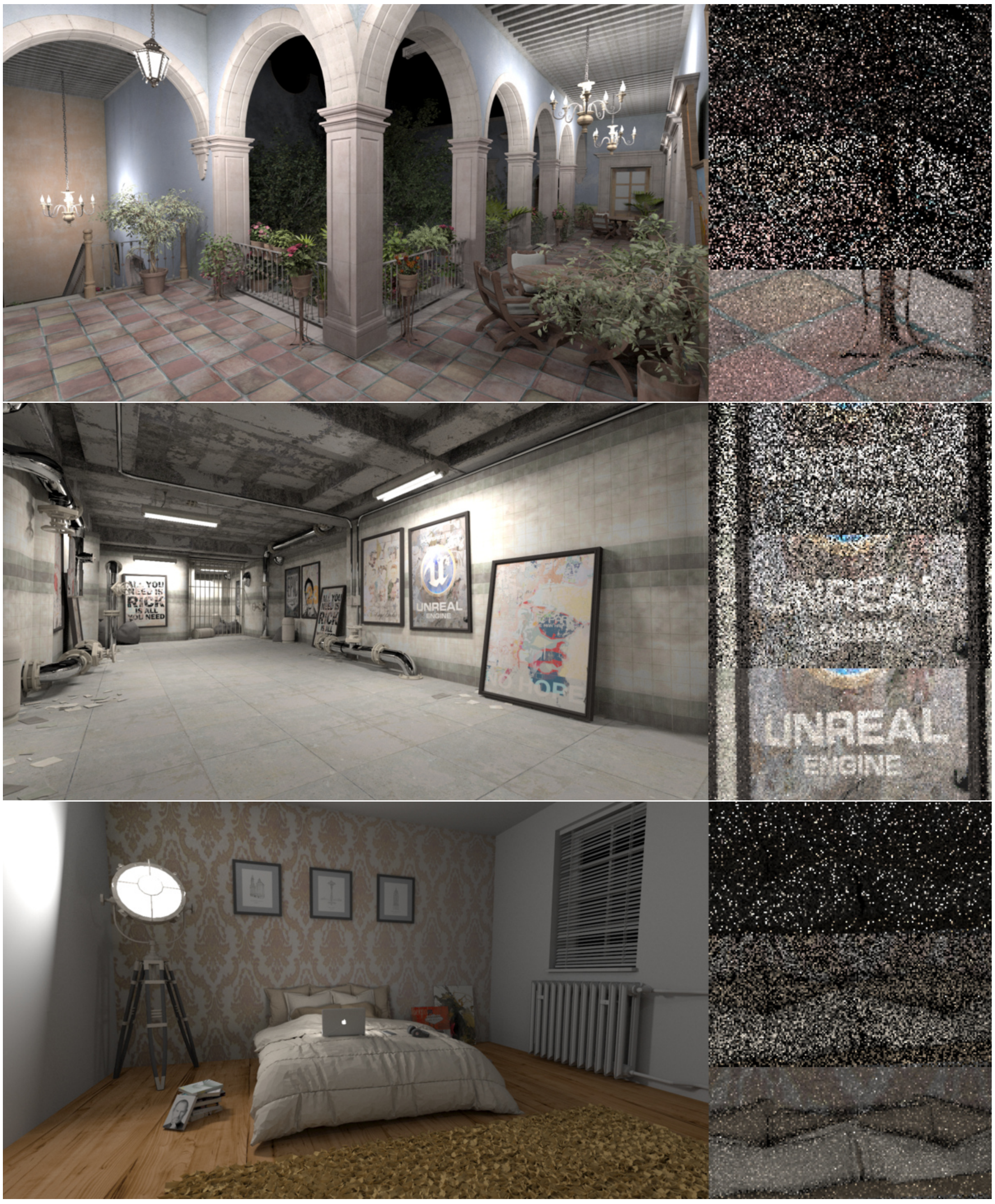


Figure 9: Algorithm comparison for different scenes. The zoomed areas have been rendered with 2 paths of length 3 per pixel and 4 shadow rays at each bounce. For each scene there are three zoomed areas. Top: Uniformly selecting one light source out of many. Middle: A light hierarchy. Bottom: Visibility-guided importance sampling. Unreal Engine demo by Epic Games.

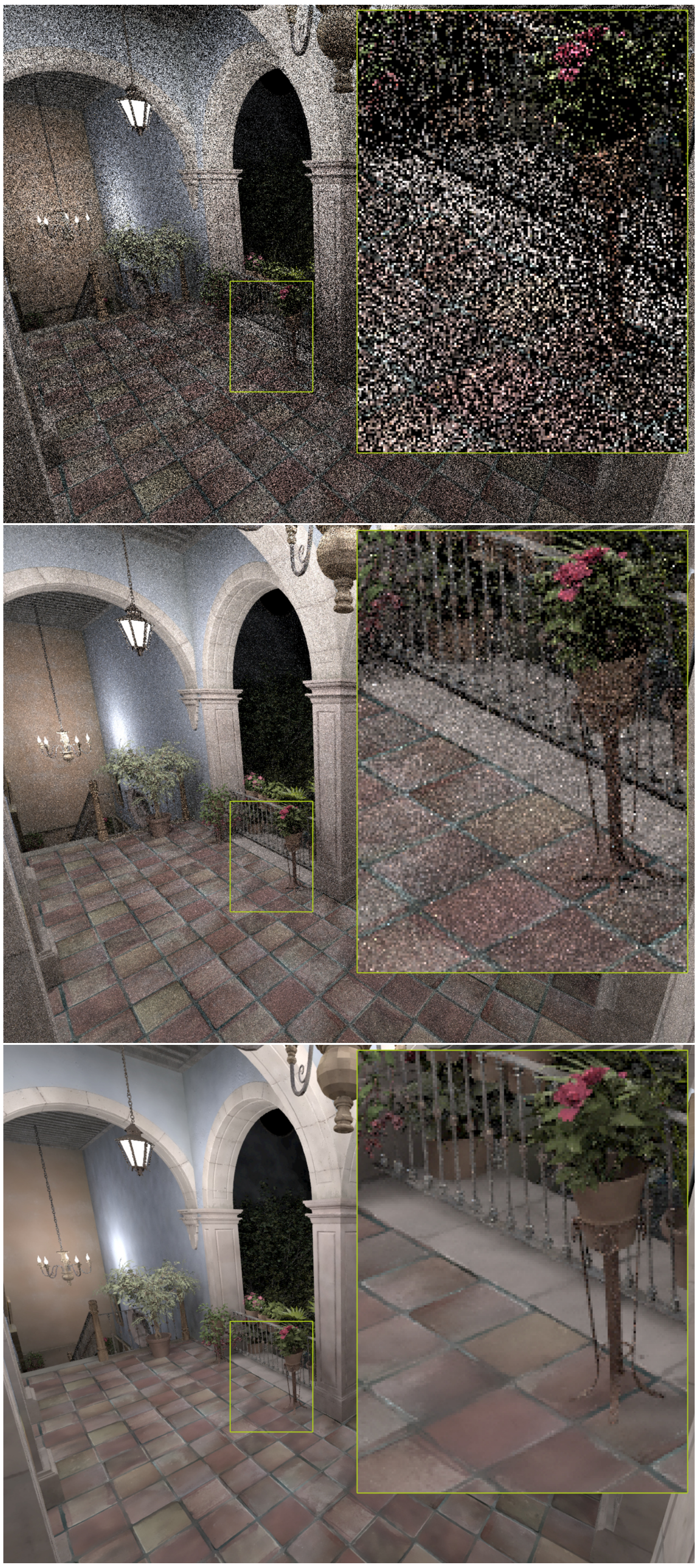


Figure 10: Comparison of a light hierarchy (top), photon-based next event estimation (middle), and a path space filtered version of the photon-based next-event estimation (bottom) at an identical budget of 4 paths of length 4 per pixel. The San Miguel model contains 10624 area light sources.

helps both temporal stability and uniform noise that is very amenable to filtering. The actual probabilities are inferred by either light paths or camera paths. The scheme is simple to implement and parallelize. It is straightforward to apply to algorithms using virtual point light sources [Keller, 1997].

While the scheme is consistent and unbiased and hence always converges to the desired solution, inefficiencies may be caused by the spatial partitioning. While a fine partitioning can capture contributions precisely, a coarse partitioning will capture contributions much faster. These are hence competing goals.

An interesting avenue for future research is to use stochastic hashing as introduced by Hachisuka [2011] and weighted reservoir sampling to avoid the construction of a variable size cumulative distribution function.

Acknowledgements

This is joint work in progress with Ken Dahm, Andreas Mikolajewski, Johannes Hanika, Nikolaus Binder, and Thomas Müller. The author is very thankful to Jiří Vorba and Chris Wyman for profound discussions. The San Miguel model is courtesy G. M. Leal LLaaguno.

References

J. Bikker. 2012. *Ray Tracing in Real-Time Games*. Ph.D. Dissertation. Delft University of Technology, the Netherlands.

N. Binder, S. Fricke, and A. Keller. 2019. Massively Parallel Path Space Filtering. *CoRR* abs/1902.05942 (2019). arXiv:1902.05942 http://arxiv.org/abs/1902.05942

N. Binder and A. Keller. 2019. Massively Parallel Construction of Radix Tree Forests for the Efficient Sampling of Discrete Probability Distributions. *CoRR* abs/1901.05423 (2019). arXiv:1901.05423 http://arxiv.org/abs/1901.05423

K. Dahm and A. Keller. 2018. Learning Light Transport the Reinforced Way. In *Monte Carlo and Quasi-Monte Carlo Methods. MCQMC 2016. Proceedings in Mathematics & Statistics*, A. Owen and P. Glynn (Eds.). Vol. 241. Springer, 181–195.

A. Estevez and C. Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 25 (Aug. 2018), 17 pages. https://doi.org/10.1145/3233305

C. Estevez and C. Kulla. 2017. Importance Sampling of Many Lights with Adaptive Tree Splitting. In *ACM SIGGRAPH 2017 Talks (SIGGRAPH '17)*. ACM, Article 33, 2 pages.

T. Hachisuka. 2011. *Robust Light Transport Simulation using Progressive Density Estimation*. Ph.D. Dissertation. University of California, San Diego.

S. Herholz, O. Elek, J. Vorba, H. Lensch, and J. Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. In *Proceedings of the Eurographics Symposium on Rendering (EGSR '16)*. Eurographics Association, 67–77. https://doi.org/10.1111/cgf.12950

H. Jensen. 1995. Importance Driven Path Tracing Using the Photon Map. In *Rendering Techniques 1995 (Proc. 6th Eurographics Workshop on Rendering)*, P. Hanrahan and W. Purgathofer (Eds.). Springer, 326–335.

H. Jensen and N. Christensen. 1995. Efficiently Rendering Shadows Using the Photon Map. In *Proc. Edugraphics + Compugraphics*, H. Santo (Ed.). GRASP- Graphic Science Promotions & Publications, 285–291.

A. Keller. 1997. Instant Radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 49–56.

A. Keller. 2013. Quasi-Monte Carlo Image Synthesis in a Nutshell. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, J. Dick, F. Kuo, G. Peters, and I. Sloan (Eds.). Springer, 203–238.

A. Keller and K. Dahm. 2019. Integral Equations and Machine Learning. *Mathematics and Computers in Simulation* abs/1712.06115 (2019).

A. Keller, K. Dahm, and N. Binder. 2016. Path Space Filtering. In *Monte Carlo and Quasi-Monte Carlo Methods 2014*, R. Cools and D. Nuyens (Eds.). Springer, 423–436.

A. Keller, C. Wächter, M. Raab, D. Seibert, D. Antwerpen, J. Korndörfer, and L. Kettner. 2017. The Iray Light Transport Simulation and Rendering System. *CoRR* abs/1705.01263 (2017). http://arxiv.org/abs/1705.01263

A. Keller and I. Wald. 2000. Efficient Importance Sampling Techniques for the Photon Map. In *Proc. VISION, MODELING, AND VISUALIZATION*. IOS Press, 271–279.

E. Lafortune and Y. Willems. 1995. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Rendering Techniques 1995 (Proc. 6th Eurographics Workshop on Rendering)*, P. Hanrahan and W. Purgathofer (Eds.). Springer, 11–20.

A. Mikolajewski. 2018. *Efficient data structures and sampling of many light sources for Next Event Estimation*. Master's Thesis. Karlsruhe Institute of Technology.

P. Moreau and P. Clarberg. 2019. Importance Sampling of Many Lights on the GPU. In *Ray Tracing Gems*, E. Haines and T. Akenine-Möller (Eds.). Apress. http://raytracinggems.com.

T. Müller, M. Gross, and J. Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. In *Proceedings of the Eurographics Symposium on Rendering*. 91–100.

T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák. 2018. Neural Importance Sampling. *CoRR* abs/1808.03856 (2018). https://arxiv.org/abs/1808.03856

M. Pharr, W. Jacob, and G. Humphreys. 2016. *Physically Based Rendering - From Theory to Implementation*. Morgan Kaufmann, Third Edition.

S. Popov, R. Ramamoorthi, F. Durand, and G. Drettakis. 2015. Probabilistic Connections for Bidirectional Path Tracing. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 34, 4 (2015). http://www-sop.inria.fr/reves/Basilic/2015/PRDD15b

I. Sobol'. 1991. *Die Monte-Carlo-Methode*. Deutscher Verlag der Wissenschaften.

J. Talbot, D. Cline, and P. Egbert. 2005. Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering (2005)*, K. Bala and P. Dutre (Eds.). The Eurographics Association, 139–146.

P. Vévoda, I. Kondapaneni, and J. Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Trans. Graph.* 37, 4, Article 125 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201340

P. Vévoda and J. Křivánek. 2016. Adaptive Direct Illumination Sampling. In *SIGGRAPH ASIA 2016 Posters*. ACM, New York, NY, USA, Article 43, 2 pages.

I. Wald, C. Benthin, and P. Slusallek. 2003. Interactive Global Illumination in Complex and Highly Occluded Environments. In *Rendering Techniques 2003 (Proc. 14th Eurographics Workshop on Rendering)*, P. Christensen and D. Cohen-Or (Eds.). 74–81.

G. Ward. 1991. Adaptive Shadow Testing for Ray Tracing. In *2nd Eurographics Workshop on Rendering*. Barcelona, Spain.

G. Ward. 1994. The RADIANCE Lighting simulation and Rendering System. In *Computer Graphics*. 459 – 472.

Q. Zheng and M. Zwicker. 2018. Learning to Importance Sample in Primary Sample Space. *CoRR* abs/1808.07840 (2018). arXiv:1808.07840 http://arxiv.org/abs/1808.07840

10 "Practical Path Guiding" in Production

Thomas Müller, *NVIDIA*§

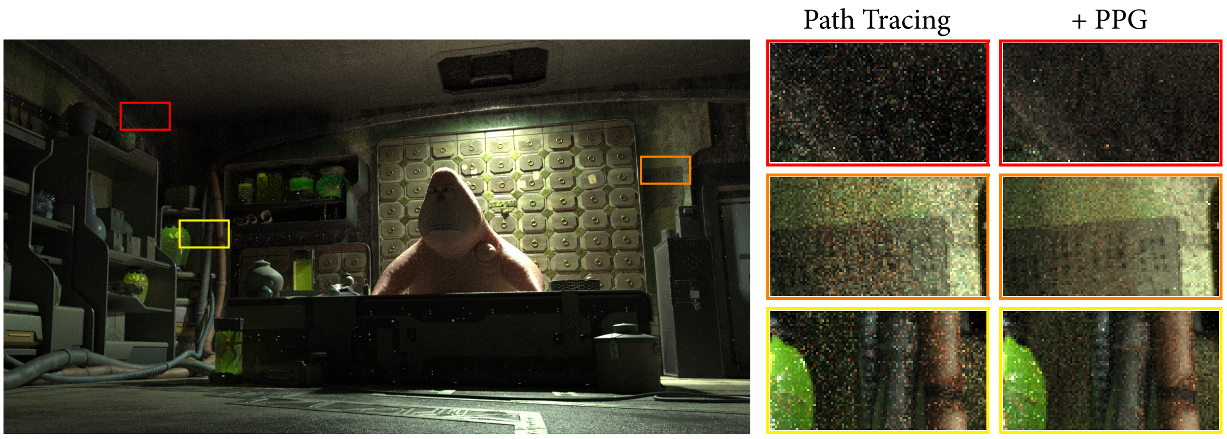


Figure 11: The "Practical Path Guiding" algorithm with our extensions reduces noise of indirect illumination in a scene lit and rendered with production assets. Copyright ©2019 Disney

10.1 Introduction

What is a "Practical" Path-Guiding Algorithm? One of the main selling points of path guiding is its ability to simulate complex light transport using a simple, low-overhead, unidirectional algorithm. This is very benefitial for production environments, because unidirectional tracing is preferred over the bidirectional methods that are typically required to simulate similar levels of complexity. However, production rendering does not always need the ability to handle complicated illumination. Many scenes encountered in production actually have quite simple light transport—in part because artists are accustomed to coping with the limitations of unidirectional tracing—which challenges the practicality of path guiding: a truly practical production-path-guiding algorithm must not only robustly handle difficult illumination, but it must also remain performant under simple illumination without introducing additional noise.

How Practical is "Practical Path Guiding"? The "Practical Path Guiding" (PPG) algorithm [Müller et al., 2017] was conceived with the goal of addressing several formerly existing practicality issues. It succeeded on some fronts, such as eliminating an expensive precomputation by instead learning to guide on-line during rendering from unidirectional camera paths (concurrently with Dahm and Keller [2018]), but on other fronts PPG was still limited: for example, its spatio-directional data structure—the SD-tree—had trouble adapting to local, high-frequency illumination, the algorithm discarded up to *half* of the total number of samples, and PPG blindly performed incident-radiance guiding everywhere, even in situations where this did not make sense.

Although these limitations did not prevent PPG from performing well under difficult illumination, they made its performance worse than that of unguided path tracing in simpler settings. Because of this, PPG clearly needed more work.

§The presented work was conducted while the author was employed at ETH Zürich and Disney Research.

Making "Practical Path Guiding" More Practical. We came up with three extensions to the original PPG algorithm that address some of the aforementioned limitations. These extensions are

1. an inverse-variance-weighted sample combination scheme that uses *most* samples as opposed to discarding up to half of them,
2. spatio-directional filtering for improving the quality and robustness of the SD-tree, and
3. on-line learning of the selection probability between guiding and BSDF sampling to avoid introducing extra noise in situations where incident-radiance guiding is suboptimal.

While there is still additional room for improvement—we do not outperform unguided path tracing in *all* situations—our extensions helped in making PPG a useful tool for movie production within Disney's Hyperion renderer [Burley et al., 2018].

In the following, we will briefly describe the original PPG algorithm and then introduce each of our extensions in detail. After that, we will discuss a number of subtle but important implementation details and conclude by stating remaining issues and open problems.

10.2 The PPG Algorithm

The ultimate goal of path guiding is to importance sample the scattering integral

$$L_\mathrm{s}(\mathbf{x}, \omega_\mathrm{o}) = \int_\mathcal{S} L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i}) f_s(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o}) \cos \gamma_\mathrm{i} \, \mathrm{d}\omega_\mathrm{i} \,, \tag{3}$$

where L_i is the spatio-directional incident radiance, f_s is the BSDF, and $\cos \gamma_\mathrm{i}$ is the foreshortening term. The PPG algorithm learns an approximation of incident radiance, denoted $\tilde{L}_\mathrm{i}$, and subsequently samples ω_i proportional to this approximation. Since this approach neglects learning the BSDF and the foreshortening term, it is paramount to combine it with BSDF sampling via multiple importance sampling (MIS) [?] to attain low variance.

At PPG's core is an iterative learning scheme: the rendering process is split into M distinct passes, called "iterations", each learning a new, more powerful incident-radiance approximation $\tilde{L}_\mathrm{i}^k$ while being guided by the approximation that was learned in the preceeding iteration $\tilde{L}_\mathrm{i}^{k-1}$. This scheme not only allows guiding early on in the rendering process—as soon as the first iteration finishes—but it also avoids an expensive precomputation that would stand in the way of fast artist workflows.

10.2.1 The SD-tree for Storing Incident Radiance

PPG approximates incident radiance as a piecewise-constant function with adaptive resolution that is represented by a spatio-directional tree (SD-tree; see Figure 12). The SD-tree has an upper part—a binary tree that subdivides the spatial domain—and a lower part—a quadtree that subdivides the directional domain. This split into two parts captures the notion that the spatial and directional domain are used in fundamentally different ways for guiding: *given* a position in space the goal is to *sample* a direction. In other words, the guiding probability distribution is *conditioned* on space and *normalized* across directions.

Recording Radiance. During path tracing, the current SD-tree $\tilde{L}_\mathrm{i}^k$ is populated with radiance estimates from paths that are guided by $\tilde{L}_\mathrm{i}^{k-1}$ as follows. Whenever a path is completed (e.g. via next-event estimation), the radiance arriving at each of its vertices v is recorded in the SD-tree leaf that contains the vertex position $\mathbf{x}_v$ and direction ω_v. This amounts to nearest-neighbor filtering of the samples as they are splatted into the tree. One of our extensions (Section 10.4) replaces this nearest-neighbor filter with a more sophisticated volume-overlap filter that significantly improves the approximation quality of the SD-tree.

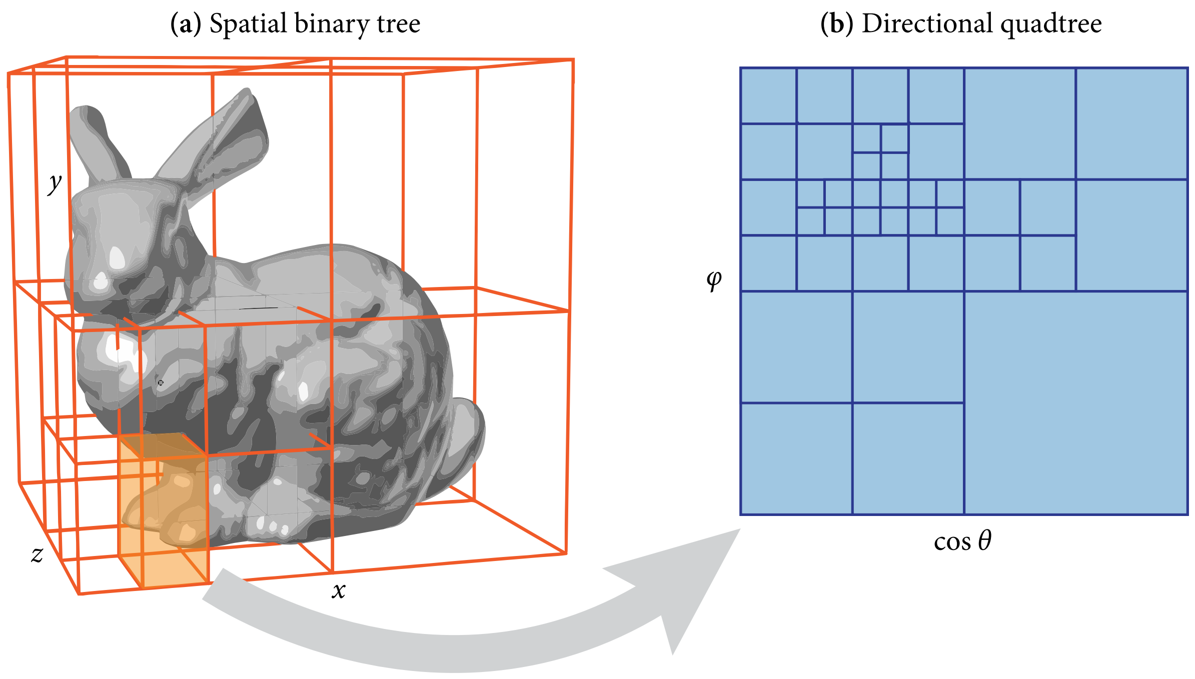


Figure 12: The SD-tree subdivides space using an adaptive binary tree (**a**) that alternates between splitting the x, y, and z axes in half. Each leaf of the spatial tree contains a quadtree (**b**), which approximates the incident radiance as an adaptively refined piecewise-constant function in cylindrical world-space coordinates $(\cos\theta, \varphi)$ with $\cos\theta = \omega^z$ and $\varphi = \operatorname{atan2}(\omega^y, \omega^x)$. Illustration from Müller et al. [2017] with additional labels.

Sampling. When using the SD-tree from the previous iteration $\tilde{L}_\mathrm{i}^{k-1}$ for importance sampling of ω_i at an intersected location $\mathbf{x}$, its spatial component is traversed to the leaf containing $\mathbf{x}$; then the quadtree contained in the leaf is sampled using hierarchical sample warping [**?**]. We provide pseudocode for sampling and for evaluating the corresponding PDF in Algorithm 2.

As mentioned before, to achieve low variance it is essential to combine SD-tree sampling with BSDF sampling via MIS: PPG probabilistically selects either SD-tree or BSDF sampling with a fixed selection probability of 50%. To reduce variance even further, in Section 10.5, we replace the 50% probability with a spatially varying value that is *optimized* jointly with the SD-tree during rendering based on the optimization of Müller et al. [2018].

Subdivision. At the end of each iteration, the newly populated SD-tree $\tilde{L}_\mathrm{i}^k$ is used to determine the subdivision of the next SD-tree $\tilde{L}_\mathrm{i}^{k+1}$ in such a way that all *spatial* leaf nodes encounter a roughly equal number of path vertices during rendering, and such that all *directional* leaf nodes of a given quadtree contain roughly the same amount of flux. We omit additional details of this subdivision process, because they are not relevant for the remainder of the text; we refer to Müller et al. [2017] and our public implementation for more details.

10.2.2 Iterative Learning and Rendering

The rendering algorithm is divided into M iterations, each of which produces an image I^k and an SD-tree $\tilde{L}_\mathrm{i}^k$. Since the early iterations are guided only by coarse, inaccurate SD-trees, they typically result in much noisier images than the later iterations. Naïvely averaging the images produced by each iteration could thus lead to more noise in the final image than simply discarding the images from early iterations. This motivates PPG's iteration scheme: rather than allocating an equal number of samples to each iteration, PPG *doubles* the number of samples of each iteration. The *total* number of samples is thus $N = 1 + 2 + \cdots + 2^{M-1} = 2^M - 1$, which is approximately twice the number of samples of the final iteration. The

Algorithm 2: Sampling and PDF evaluation of the directional quadtree.

```
function sampleQuadtree(node)
    if isLeaf(node) then
        return cylindricalToDirection(uniformRandomPositionIn(node))
    else
        childNode ← sampleChildByEnergy(node)
        return sampleQuadtree(childNode)

function pdfQuadtree(node, $\omega_\mathrm{i}$)
    if isLeaf(node) then
        return $1/4\pi$
    else
        childNode ← getChild(node, directionToCylindrical($\omega_\mathrm{i}$))
        $\beta$ ← 4 · getFlux(childNode) / getFlux(node)
        return $\beta$ · pdfQuadtree(childNode, $\omega_\mathrm{i}$)
```

algorithm then discards the images produced by all but the final iteration, preventing initial high-variance samples from increasing overall noise while limiting the "wasted" computation to at most half of the total sample count.[5]

While this scheme increases robustness in the worst cases, it is clearly undesirable to throw away half of the computation when the initial iterations already resulted in small variance. Müller et al. [2017] address this problem by adaptively assigning a larger proportion of samples to the final iteration, thereby discarding a smaller proportion, but the heuristic they use for this purpose is brittle in practice. In the next section, we explain an alterative: a principled weighting scheme for combining almost all samples in a robust manner.

10.3 Extension 1: Inverse-Variance-Weighted Sample Combination

Our first extension allows robustly combining the majority of samples by using an inverse-variance-based weighting scheme that acts on the images $I^1, I^2, \ldots, I^M$ produced in each iteration.

Theoretically Optimal Sample Combination. Because the images $I^1, I^2, \ldots, I^M$ are independent random variables, the *optimal* per-pixel combination weights—i.e. those that result in the least combined variance—are proportional to the *inverse pixel variance* [?]

$$I(p) = \frac{1}{\sum_{i=1}^{M} w^k(p)} \sum_{i=1}^{M} w^k(p) I^k(p)\,, \tag{4}$$

$$w^k(p) = \frac{1}{\mathbb{V}[I^k(p)]}\,, \tag{5}$$

where $I(p)$ is the combined pixel value of pixel p.

Robust Sample Combination. Unfortunately, it is unrealistic to assume accurate knowledge of the variance of each individual pixel, which makes the aforementioned optimal scheme difficult to apply. While it *is* possible to estimate the pixel variance from the samples themselves, such estimates often are highly inaccurate and would therefore lead to suboptimal weights if they were used. But even worse: estimating the variance from the samples introduces *correlation* between the image I^k and the variance-estimate-based weights $w^k(p)$, ultimately leading to bias.

We want to reduce this bias and increase stability while retaining the core idea behind the inverse-variance sample combination. To this end, we propose to average the per-pixel variance estimates over

[5]The algorithm can also handle non-power-of-two sample counts by *lengthening* the last iteration.

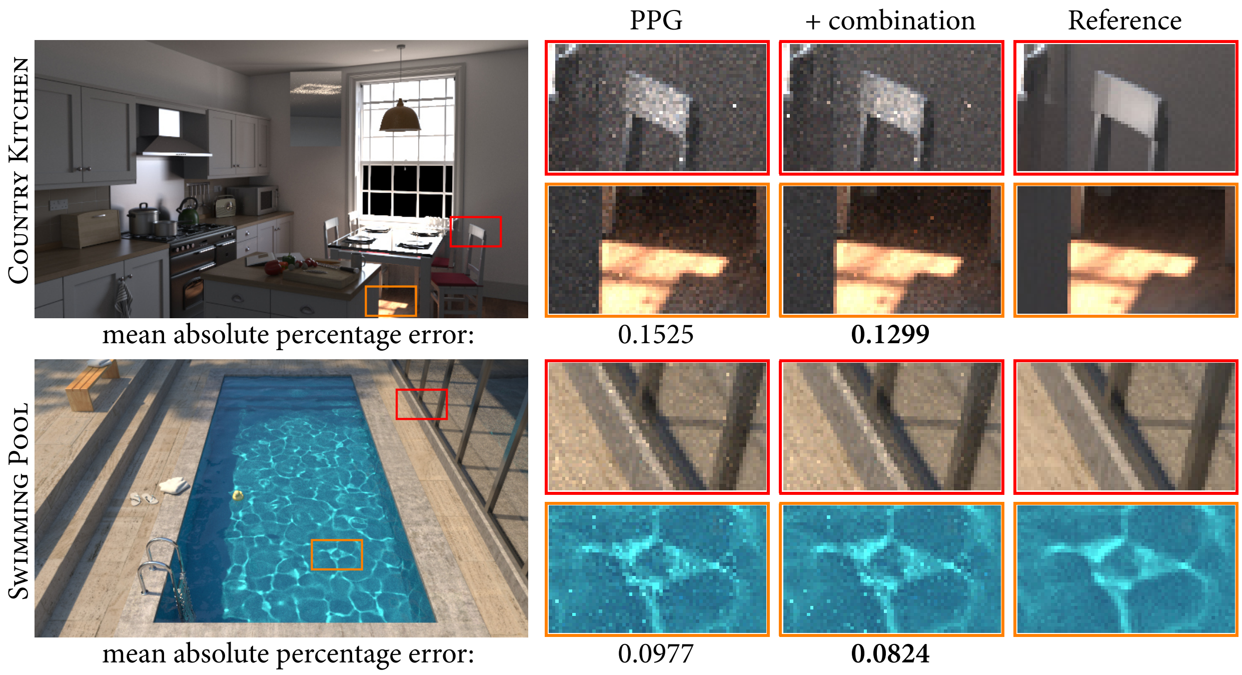


Figure 13: Combining the images produced by each iteration weighted by their mean pixel variance reduces the number of "wasted" samples and thereby slightly subdues noise. We quantify the noise as *mean absolute percentage error* averaged over each image.

the image plane

$$\bar{w}^k = \sum_p w^k(p)\,, \tag{6}$$

leading to the less biased but also less optimal whole-image weighting scheme

$$I = \frac{1}{\sum_{i=1}^{M} \bar{w}^k} \sum_{i=1}^{M} \bar{w}^k I^k\,. \tag{7}$$

In our experiments, we were unable to actually observe any bias using this scheme, both visually and numerically. Furthermore, the resulting images consistently had less noise compared to images produced by the original PPG algorithm; we show two examples in Figure 13.

Importantly, the proposed weighting scheme is *consistent*. As rendering progresses, the variance estimate is built using an increasing number of samples, approaching the true variance of the pixel value. This higher accuracy of the variance estimate leads to vanishing correlation between the weights and the pixel values—because the true variance does not depend on any specific samples—and thereby to vanishing bias.

Lastly, to further improve robustness, we only combine the *last* 4 images. While this amounts to always discarding slightly fewer than the first 6.25% of the samples, these initial samples are the noisiest and can—in the most difficult cases—otherwise cause unstable weights.

10.4 Extension 2: Spatio-directional Splatting into the SD-Tree

While testing the SD-tree on a large number of scenes, we observed distracting artifacts under spatio-directionally narrow illumination. In Figure 14, we created a contrived situation that demonstrates the problem: we render a Cornell Box that is illuminated by a tiny quad light with *disabled* next-event estimation. Although PPG dramatically improves overall variance over an unguided path tracer (left), residual noise manifests non-uniformly along the spatial structure of the SD-tree (middle).

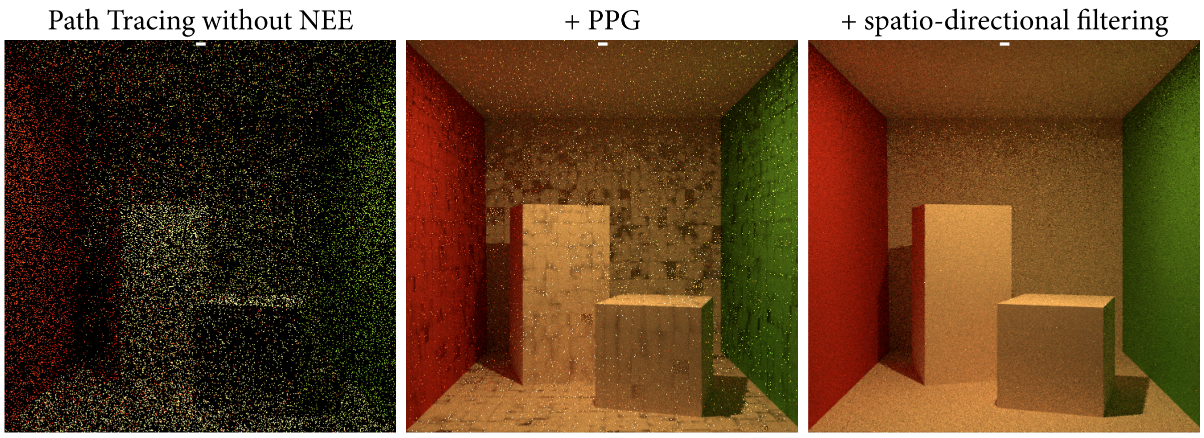


Figure 14: Improved handling of narrow illumination by spatio-directional filtering of radiance estimates that are splatted into the SD-tree. We illustrate a contrived situation: a small light source that is *not* sampled with next-event estimation (NEE). Although PPG (middle) dramatically improves overall variance compared to an unguided path tracer (left), residual noise is distributed non-uniformly along the spatial structure of the SD-tree, which is visually unpleasing. Spatio-directional filtering (right) addresses the structured noise by distributing it evenly across the scene and at the same time reducing it overall.

These artifacts are primarily caused by three problems of the SD-tree: first, the spatial subdivision scheme assumes that splitting a node in half causes the newly created nodes to receive a roughly equal number of samples, which is not the case when path vertices are distributed anisotropically, e.g. along geometric edges. Second, even *if* the path vertices were evenly distributed across spatial leaf nodes, their sample variance is disregarded by the algorithm. And lastly, during path tracing, incident radiance estimates at path vertices are recorded within their *nearest* spatial leaves, which causes the learned approximation to be better in the center of nodes than at their edges, leading to visible darkening at leaf boundaries.

All the above problems result in undesired *non-uniform* learning of the SD-tree, giving rise to the artifacts in Figure 14 (middle). A popular and effective approach to mitigate such non-uniformity is filtering. We therefore introduce a spatio-directional filter to the splatting of radiance estimates into the SD-tree: instead of recording radiance estimates $\langle L_\mathrm{i} \rangle$ from vertices v within the leaf node that contains the vertex position $\mathbf{x}_v$ and direction ω_v, we record $\langle L_\mathrm{i} \rangle$ within those leaf nodes that fall into a *spatial neighborhood* around $\mathbf{x}_v$ and a *directional neighborhood* around ω_v.

More concretely, given $\langle L_\mathrm{i} \rangle$ at a vertex v, we begin by traversing the spatial tree to obtain the *spatial footprint* and corresponding volume V of the leaf containing $\mathbf{x}_v$; this footprint determines the filter radius and thereby the size of the spatial neighborhood. We then traverse the spatial tree again, this time visiting each node that has non-zero volume overlap with the spatial footprint from before, centered around $\mathbf{x}_v$. For each spatial leaf with non-zero volume overlap V_o that we find this way, we record the radiance estimate weighted by the fraction of overlapped volume $\langle L_\mathrm{i} \rangle \cdot V_o/V$.

Directional filtering works analogously, only that we perform area-based filtering over the cylindrical domain as opposed to spatial filtering over space.

Stochastic Filtering. When implemented as described above, the filtering operation comes with significant computational cost. This cost can be mostly avoided at the expense of slightly worse quality by replacing the deterministic filtering that traverses entire sub-trees with stochastic filtering that traverses only towards a single leaf: after obtaining the spatial footprint, the position $\mathbf{x}_v$ is *jittered* (i.e. positioned uniformly at random) within the footprint's volume—again, centered around $\mathbf{x}_v$—and then recorded in the SD-tree as done in the original algorithm without weighting the radiance by V_o/V.

To find the sweet-spot between quality and computational cost, it is important to consider the compounding effect of enabling deterministic spatial *and* directional filtering at the same time: since deterministic spatial filtering increases the number of visited quadtrees (from one to many) and directional

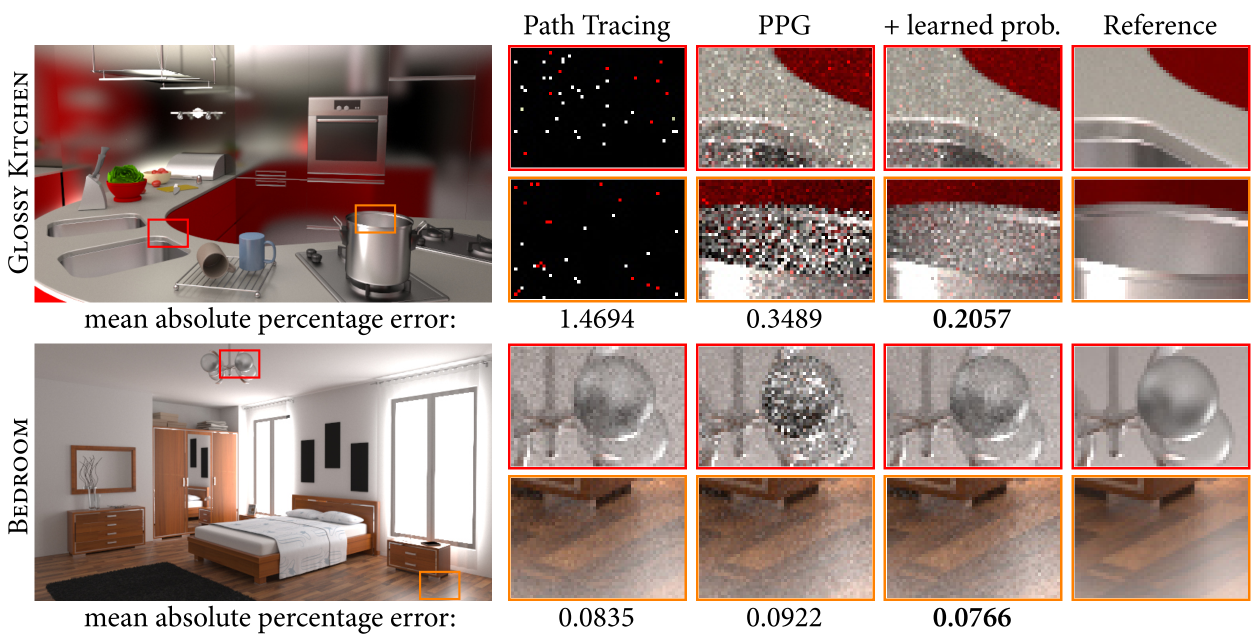


Figure 15: Learning the selection probability of multiple importance sampling reduces noise in difficult glossy light transport (top) and prevents path guiding from producing overall much worse results in scenes that are simple to render using BSDF sampling alone (bottom). We use 1024 samples per pixel and we quantify noise using *mean absolute percentage error* averaged over each image.

filtering increases the number of visited leaf quads *for each visited quadtree* (again, from one to many), the total computational cost is roughly proportional to the *product* of the spatial filtering overhead and the directional filtering overhead. It follows, that negating the overhead via stochastic filtering on only *one* of the two filtering operations provides *most* of the possible speedup.

Additionally, since spatial filtering operates on 3 dimensions whereas directional filtering operates only on 2, the overhead caused by spatial filtering is bigger.

Because of these two reasons, we perform spatial filtering stochastically and use deterministic filtering only in the directional domain. The computational overhead of this combined approach over vanilla PPG is around 20% in a Mitsuba-rendered CORNELL BOX and below 10% in Hyperion-rendered production scenes. The smaller overhead in Hyperion is a consequence of more computation being devoted to rendering significantly more complex scenes.

Subdivision Criterion. The original PPG algorithm subdivides spatial leaf nodes when they receive more than $c = 12000 \cdot 2^{k/2}$ samples. Due to the increased robustness from filtering, we are able to reduce this number to $c = 4000 \cdot 2^{k/2}$, thereby leading to a finer subdivision and therefore not only a more robust but also a more fine-grained fit; see Figure 14.

10.5 Extension 3: Optimization of MIS Selection Probability

Our third extension aims at on-line learning of the multiple-importance-sampling selection probability between SD-tree and BSDF sampling such that variance is minimized. This addresses the problem of overusing the SD-tree on glossy surfaces and when it approximates incident radiance poorly. We demonstrate the benefits of this extension in Figure 15.

Objective Definition. Recall that in PPG, radiance sampling is combined with BSDF sampling using the *one-sample* MIS model [**?**]. Mathematically, this amounts to sampling according to a linear combination of the SD-tree PDF $q \propto \tilde{L}_\mathrm{i}$ and the BSDF PDF $p_{f_s} \propto f_s$

$$\hat{q}(\omega_\mathrm{i}|\mathbf{x}, \omega_\mathrm{o}; \alpha) = \alpha \cdot p_{f_s}(\omega_\mathrm{i}|\mathbf{x}, \omega_\mathrm{o}) + \big(1 - \alpha\big) \cdot q(\omega_\mathrm{i}|\mathbf{x}) \,, \tag{8}$$

where α is the *selection probability* that determines how frequently each of the two strategies is sampled from. A value of 0 amounts to always sampling from the SD-tree, whereas a value of 1 corresponds to always using BSDF sampling.

Many radiance-based guiding approaches use a fixed value of $\alpha = 0.5$ [Müller et al., 2017, Vorba et al., 2014], whereas others that learn the product are typically more aggressive: Herholz et al. [2016], **?** use $\alpha = 0.1$. However, it would be better to let α vary spatially to account for the fact that BSDF sampling may be more suitable in some scene regions, whereas guiding may be more appropriate in others.

With this motivation in mind, our objective is to replace the fixed selection probability α with a *learned* function $\alpha(\mathbf{x})$. To derive a learning algorithm for $\alpha(\mathbf{x})$, we begin by formalizing the desired form that we would like the combined PDF $\hat{q}$ to take.

Zero variance can only be attained when sampling proportional to the product of incident radiance and the BSDF, i.e. according to the "ideal" PDF $p(\omega_\mathrm{i}|\mathbf{x}, \omega_\mathrm{o}) \propto L_\mathrm{i}(\mathbf{x}, \omega_\mathrm{i}) f_s(\mathbf{x}, \omega_\mathrm{i}) \cos\gamma_\mathrm{i}$. Although it is impossible to *perfectly* match the ideal PDF simply by varying $\alpha(\mathbf{x})$, our goal is to optimize $\alpha(\mathbf{x})$ such that the combined PDF $\hat{q}$ at least approximates the ideal PDF p as closely as possible.

Since the goal of approximating $p(\omega_\mathrm{i}|\mathbf{x}, \omega_\mathrm{o})$ with $\hat{q}(\omega_\mathrm{i}|\mathbf{x}, \omega_\mathrm{o})$ can be accomplished independently for any spatio-directional coordinate $(\mathbf{x}, \omega_\mathrm{o})$, we will omit $\mathbf{x}$ and ω_o from the following derivations. The objective is then to approximate $p(\omega_\mathrm{i})$ with $\hat{q}(\omega_\mathrm{i}; \alpha) = \alpha \cdot p_{f_s}(\omega_\mathrm{i}) + (1 - \alpha) \cdot q(\omega_\mathrm{i})$ by varying α. We can formalize this as an optimization problem: we try to minimize a suitably chosen *distance* $D(p \,\|\, \hat{q}; \alpha)$ between the PDFs p and $\hat{q}$, referred to as "objective function". The optimal selection probability is then the one that minimizes the objective function

$$\hat{\alpha} = \operatorname*{arg\,min}_{\alpha \in [0,1]} D(p \,\|\, \hat{q}; \alpha) \,. \tag{9}$$

Choice of Objective Function. Ideally, we would set the objective function D to the Monte Carlo variance, such that the above equation directly corresponds to minimizing variance. Such an optimization *is* actually feasible, but it is numerically unstable which leads to worse results compared to alternative objectives [Müller et al., 2018]. We therefore use another, more numerically stable function to capture the difference between p and $\hat{q}$: the Kullback-Leibler (KL) divergence.

The KL divergence between the ideal PDF p and the learned PDF $\hat{q}$ is defined as

$$D_\mathrm{KL}(p \,\|\, \hat{q}; \alpha) = \int_\mathcal{S} p(\omega_\mathrm{i}) \log \frac{p(\omega_\mathrm{i})}{\hat{q}(\omega_\mathrm{i}; \alpha)} \, \mathrm{d}\omega_\mathrm{i} \,. \tag{10}$$

It is a good surrogate for the variance, because—like the variance—it attains a value of 0 if and only if $p = \hat{q}$ and because it approaches infinity as $\hat{q}$ undersamples the integrand, i.e. when $\hat{q}(\omega_\mathrm{i}; \alpha)$ approaches zero for directions where $p(\omega_\mathrm{i}) > 0$.

10.5.1 Minimizing the Kullback-Leibler Divergence

Müller et al. [2018] showed that the selection probability α can be optimized such that the KL divergence $D_\mathrm{KL}(p \,\|\, \hat{q}; \alpha)$ is minimized when α is the output of a neural network. In this section, we use an adapted approach that does the same *without* involving a neural network.

We minimize the KL divergence by setting its gradient to zero. Ideally, we would do this in closed form, which is unfortunately not possible because of the difficult integral. Such difficult minimization problems are well studied in the field of machine learning and are often addressed using algorithms that build on "stochastic gradient descent". One of the most successful of such approaches is "Adam" [**?**], which is remarkably simple to implement. For this reason, and for another reason that we will mention shortly, we use Adam to optimize the selection probability α.

The Adam algorithm takes stochastic estimates of the "loss gradient" (in our case: the gradient of the KL divergence) as input and is guaranteed to converge to a (locally) optimal value if these estimates are *unbiased*. Fortunately, it *is* possible to obtain unbiased KL-divergence gradient estimates: Müller et al. [2018] showed, that the gradient of the KL divergence can be written as an expectation over samples drawn from an arbitrary PDF q_s.[6]

$$\nabla_\alpha D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) = \mathop{\mathbb{E}}_{\omega_i \sim q_s}\left[-\frac{p(\omega_i)}{q_s(\omega_i)} \nabla_\alpha \log \hat{q}(\omega_i; \alpha) \right], \tag{11}$$

which yields the desired unbiased gradient as the inner part of the above expectation for directions ω_i drawn from q_s.

Evaluating this unbiased gradient is difficult, because we cannot evaluate the ideal PDF $p(\omega_i)$ in closed form. We can, however, replace the ideal PDF with an unnormalized unbiased estimate of it: the product of incident radiance, the BSDF, and the foreshortening term. This replacement introduces a constant factor but otherwise leaves the unbiasedness of the gradient intact because of the linearity of expectations. Slightly abusing notation and denoting $\langle X \rangle$ as an *unbiased estimate* of X, we have

$$c \cdot \langle \nabla_\alpha D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) \rangle = -\frac{\langle L_i(\omega_i) \rangle f_s(\omega_i) \cos\gamma_i}{q_s(\omega_i)} \nabla_\alpha \log \hat{q}(\omega_i; \alpha)\,. \tag{12}$$

Here lies the second reason behind our choice of using the Adam optimizer: Adam automatically compensates for the constant factor c, so we ignore the factor and express the gradient only up to proportionality.

We can further expand the gradient estimate by applying the chain rule, finally leading to an expression that can be evaluated within a renderer

$$\begin{aligned}\langle \nabla_\alpha D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) \rangle &\propto -\frac{\langle L_i(\omega_i) \rangle f_s(\omega_i) \cos\gamma_i}{q_s(\omega_i)} \nabla_\alpha \log \hat{q}(\omega_i; \alpha) \\ &= -\frac{\langle L_i(\omega_i) \rangle f_s(\omega_i) \cos\gamma_i}{q_s(\omega_i)} \frac{\nabla_\alpha \hat{q}(\omega_i; \alpha)}{\hat{q}(\omega_i; \alpha)} \\ &= -\frac{\langle L_i(\omega_i) \rangle f_s(\omega_i) \cos\gamma_i}{q_s(\omega_i)} \frac{p_{f_s}(\omega_i) - q(\omega_i)}{\hat{q}(\omega_i; \alpha)}\,. \end{aligned} \tag{13}$$

It may be tempting to directly feed this gradient into the Adam algorithm to optimize the selection probability α, but there is a remaining problem we must solve before we can do so: the selection probability is only valid in the range $[0, 1]$, but the above optimization is unaware of this constraint. The optimization could therefore produce invalid probabilities outside of $[0, 1]$. We *enforce* probabilities that lie within $[0, 1]$ by modeling α in terms of an auxiliary "latent" variable $\theta \in \mathbb{R}$ that can take *any* real value without constraint, using the logistic sigmoid function $\sigma : \mathbb{R} \to [0, 1]$

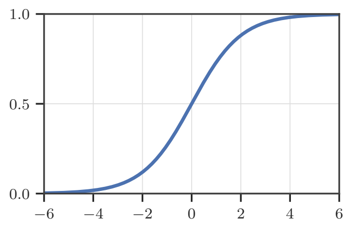


Figure 16: Logistic sigmoid.

$$\alpha(\theta) \equiv \sigma(\theta) = \frac{1}{1 + e^{-\theta}}\,. \tag{14}$$

Since the sigmoid enforces valid probabilities α, we can optimize θ using Adam without worrying about the range of α. To perform this optimization, we need an unbiased gradient estimate w.r.t. θ instead of α, which can be obtained using the chain rule

$$\begin{aligned}\langle \nabla_\theta D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) \rangle &= \langle \nabla_\alpha D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) \rangle \cdot \nabla_\theta\, \alpha(\theta) \\ &= \langle \nabla_\alpha D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) \rangle \cdot \alpha(\theta)\big(1 - \alpha(\theta)\big) \\ &\propto -\frac{\langle L_i(\omega_i) \rangle f_s(\omega_i) \cos\gamma_i}{q_s(\omega_i)} \frac{p_{f_s}(\omega_i) - q(\omega_i)}{\hat{q}(\omega_i; \alpha)} \alpha(\theta)\big(1 - \alpha(\theta)\big)\,. \end{aligned} \tag{15}$$

[6]In practice, q_s is chosen to equal $\hat{q}$, but implementation details of multithreaded optimization necessitate their distinction.

Algorithm 3: Optimization of the MIS selection probability. One MIS optimization step is executed whenever a radiance estimate is splatted into a spatial leaf of the SD-tree.

```
class SpatialLeaf()
    $t, m, v, \theta \leftarrow 0$                                          // Initialize state
    $\beta_1 \leftarrow 0.9$, $\beta_2 \leftarrow 0.999$, $\varepsilon \leftarrow 10^{-8}$, learningRate $\leftarrow 0.01$, regularization $\leftarrow 0.01$   // Hyperparameters
    function adamStep($\nabla_\theta$)
        $t \leftarrow t + 1$                                                // Increment iteration counter
        $l \leftarrow$ learningRate $\cdot \sqrt{1 - \beta_1^t}/(1 - \beta_2^t)$          // Compute de-biased learning rate
        $m \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta$                  // Update first moment
        $v \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla_\theta \cdot \nabla_\theta$          // Update second moment
        $\theta \leftarrow \theta - l \cdot m/(\sqrt{v} + \varepsilon)$                       // Update parameter
    function misOptimizationStep($\mathbf{x}$, $\omega_\mathrm{i}$, $\omega_\mathrm{o}$, radianceEstimate, samplePdf)
        productEstimate $\leftarrow$ radianceEstimate $\cdot f_s(\omega_\mathrm{i}) \cos \gamma_\mathrm{i}$
        bsdfPdf $\leftarrow p_{f_s}(\omega_\mathrm{i} | \omega_\mathrm{o}, \mathbf{x})$
        learnedPdf $\leftarrow$ isDiscrete($f_s(\omega_\mathrm{i})$) ? 0 : $q(\omega_\mathrm{i} | \mathbf{x})$
        spin lock (this)                          // Ensure $\theta$ is only optimized by one thread at a time
            $\alpha \leftarrow$ selectionProbability()
            combinedPdf $\leftarrow \alpha \cdot$ bsdfPdf $+ (1 - \alpha) \cdot$ learnedPdf         // Equation 8
            $\nabla_\alpha \leftarrow -$productEstimate $\cdot$ (bsdfPdf $-$ learnedPdf)/(samplePdf $\cdot$ combinedPdf)   // Equation 13
            $\nabla_\theta \leftarrow \nabla_\alpha \cdot \alpha(1 - \alpha)$                              // Chain rule
            regGradient $\leftarrow$ regularization $\cdot \theta$          // L2 regularization to avoid sigmoid saturation
            adamStep($\nabla_\theta$ + regGradient)
    function selectionProbability()          // Called by the path tracer to use the learned probability
        return $1/(1 + e^{-\theta})$                                    // Sigmoid as in Equation 14
```

10.5.2 Integration of MIS Optimization into PPG

In this section, we describe how we implemented the selection-probability optimization using Adam within the PPG algorithm.

Spatial Discretization. Our goal is to optimize the selection probability α spatially and jointly with the learning of the SD-tree. To this end, we utilize the spatial subdivision of the SD-tree: in each spatial leaf node, we not only store a directional quadtree, but also the latent variable θ controlling the selection probability α. During rendering, whenever we splat a radiance estimate into a spatial leaf node, we not only record it within the corresponding quadtree, but we also execute an Adam optimization step.

Multi-Threading. The gradient computation and Adam optimization are not thread-safe, so mutual exclusivity must be guaranteed when two threads attempt to perform an optimization step within the same spatial leaf concurrenty. Fortunately, such thread collisions are rare, because the spatial binary tree has a resolution that approximately matches the spatial density of path vertices. We are therefore able to use an inexpensive spin lock, with which we observed near-perfect linear performance scaling up to 48 threads, which is the maximum number we could test on.

Regularization. Recall, that we express the selection probability as the sigmoid of a latent variable $\alpha(\theta) \equiv \sigma(\theta)$. This sigmoid levels off as θ approaches positive or negative infinity (see Figure 16), i.e. its gradient approaches zero. Unfortunately, small gradients make gradient-based optimizers such as Adam less effective, which is known in machine learning as the "vanishing-gradient" problem.

We limit the vanishing-gradient problem by introducing L^2 regularization to our latent variable θ, which "encourages" the optimization to prefer values of θ that are closer to zero, i.e. values of α that are closer to 0.5. This amounts to modifying our objective function (the KL divergence) to include an additive term $\lambda\theta^2$, where λ controls the strength of the regularization. The modified gradient estimate is then

$$\langle\nabla_\theta(D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha) + \lambda\theta^2)\rangle = \langle\nabla_\theta D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha)\rangle + \nabla_\theta\lambda\theta^2 = \langle\nabla_\theta D_{\mathrm{KL}}(p \,\|\, \hat{q}; \alpha\rangle + 2\lambda\theta\,. \tag{16}$$

We pick a weak regularization $\lambda = 0.005$, allowing the optimization to produce selection probabilities close to 0 or 1 (e.g. 0.01 or 0.99), but not close enough that the optimization suffers.

Discrete-Smooth Hybrid BSDFs. The above optimization can handle BSDFs that are hybrids of smooth and discrete reflectances. Abusing mathematics a little bit, whenever a discrete component is sampled, the smooth SD-tree PDF $q(\omega_\mathrm{i})$ must be treated as zero[7] (see Algorithm 3), similar to how smooth BSDF components are usually treated as zero in such cases.

Code. We provide pseudocode of the *full* MIS optimization—including Adam—in Algorithm 3. The pseudocode also includes the implementation details that we discussed in the preceeding paragraphs. For an actual implementation within the Mitsuba renderer, we refer to our public code at https://github.com/Tom94/practical-path-guiding.

10.6 Adjoint-Driven Russian Roulette and Splitting

Russian roulette is an important building block for making almost any path tracer efficient. Unfortunately, path-throughput-based russian roulette typically has detrimental effects on guided path tracers, because the throughput does not account for incident illumination (the adjoint) encountered upon path completion. While it is possible to avoid such detrimental effects by entirely disabling russian roulette, this comes with potentially significant extra computational cost.

Adjoint-driven russian roulette (and splitting) [Vorba and Křivánek, 2016] solves this issue by incorporating an estimate of the adjoint—the learned incident-radiance approximation $\tilde{L}_\mathrm{i}$—into the russian-roulette decision. We use this scheme because of its increased efficiency and because of its easy implementation within a path-guiding algorithm that learns incident radiance. For details on adjoint-driven Russian roulette and splitting we refer to Sec. 7.7.

10.7 Miscellaneous Implementation Details

In the following, we discuss miscellaneous implementation details of the PPG algorithm that are easy to overlook but greatly improve its effectiveness.

Next-Event Estimation. Although path guiding is able to sample complex *general* illumination, it often performs worse than next-event estimation (NEE) on *direct* illumination, especially when NEE uses an importance cache such as the one used in Hyperion [Burley et al., 2018]. Because of this, we enable NEE and do not train the SD-tree with direct illumination from light sources that are sampled by NEE. The SD-tree is therefore trained using *all* indirect illumination and only direct illumination from light sources that are *not* sampled by NEE (e.g. emissive volumes).

Parameterization of Directional Distribution. There are a number of possible ways to parameterize the directional guiding distribution. We use *world-space-aligned cylindrical coordinates* for two practical reasons. First, world-space alignment—as opposed to surface alignment—allows the usage of the same distribution of incident radiance when there is high-frequency normal variation. Second, we use cylindrical coordinates—as opposed to spherical coordinates—due to their area-preserving correspondence to the surface of the solid sphere.

[7]Most guiding methods—including PPG—do not learn discrete components. However, if a discrete component *was* learned, then it should *not* be treated as zero in this situation.

Rapid Adaptation of the SD-tree to Large Scenes. It is common in production to encounter vast scenes of which only a small portion is visible to the camera. For a path-guiding data structure to be effective in such situations, it must rapidly adapt to the distribution of paths being traced. Although the SD-tree *does* continually adapt to the distribution of paths, the amount of adaptation *within each iteration* is relatively small. To address this problem, we perform a small number of 1-sample-per-pixel iterations (e.g. 8) at the beginning of rendering. This has the effect of "initializing" the subdivision of the SD-tree to match the camera frustum before it is used for path guiding in the remaining rendering process.

10.8 Remaining Issues and Future Work

In this section, we mention several known issues of PPG that we were not able to address or did not have time to investigate yet. These remaining issues provide interesting directions of future work, both within the context of PPG as well as in the pursuit of replacing PPG (or components of it) with better alternatives.

Subdivision of the SD-tree. In Section 10.4, we proposed to combat non-uniform learning of the SD-tree using spatio-directional filtering. While this approach is effective at reducing the symptoms of the problem—i.e. a bad, non-uniform incident-radiance approximation—it does not tackle its fundamental causes, which are rooted in PPG's suboptimal subdivision scheme. In the future it is worth investigating alternative data structures such as the BSP-tree proposed by Herholz et al. [2019] (described in detail in Section 11.5.1) or neural networks that internally learn a spatio-directional representation [Müller et al., 2018].

Product Guiding. The SD-tree can only learn to guide according to incident radiance, which limits its practicality compared to alternative approaches that can guide according to the full product [Müller et al., 2018, Herholz et al., 2016, 2019, ?]. Although it *is* possible to sample from the product of the SD-tree and a BSDF represented by Haar wavelets [?] or spherical harmonics [?], such representations are difficult to obtain for rich, parametric BSDFs such as the Disney BSDF [?] which is used in the Hyperion renderer.

Temporal Guiding. The guiding of motion-blur effects is difficult with PPG, because the motion can cause large incident-radiance variation that is not captured by the SD-tree. As Müller et al. [2017] already suggested, it may be possible to simply add the temportal dimension to the spatial binary tree (making it a 4-D spatio-temporal binary tree), but this is yet to be tested.

Volumetric Guiding. PPG performs poorly when used to guide volumetric path tracing. This is because of a number of reasons such as the lack of product guiding with the phase function and the lack of guided distance sampling. Progress towards volumetric path guiding has been made by Herholz et al. [2019] using a spatial data structure similar to our binary tree in combination with a directional parametric mixture model. Sebastian Herholz will discuss more details in Section 11.5.

Simple Light Transport. Lastly, there is still room for improving PPG under easy-to-render illumination. Even though our extensions achieve better results than vanilla PPG in difficult scenes (Figure 15, top) and surpass unguided path tracing in some simple settings (e.g. in Figure 15, bottom), there are remaining cases where unguided path tracing is superior. These are caused mostly by the 10–20% computational overhead of the SD-tree and the discarding of all but the last 4 PPG iterations (around 6.25% of all samples). It is therefore important future work to further improve upon the algorithm and to investigate the possibility of effective guiding of *direct* illumination that complements next-event estimation for better overall efficiency.

10.9 Conclusion

Our goal was to obtain a path-guiding algorithm that not only accelerates rendering of difficult light transport, but also performs no worse than simple unidirectional path tracing under simple illumination.

To this end, we reviewed the "Practical Path Guiding" algorithm [Müller et al., 2017] and introduced three extensions that helped us improve its effectiveness and robustness. First, we avoided discarding large fractions of the samples by weighting them approximately proportional to their inverse variance. Next, we improved the quality of the learned SD-tree by spatio-directional filtering of splatted radiance estimates, and lastly, we showed how the MIS selection probability can be optimized to increase the overall efficiency of guiding.

After that, we discussed a number of important details to consider when implementing PPG with our extensions in a production environment. These include the combination of PPG with next-event estimation, the use of adjoint-driven russian roulette, and various aspects of effectively utilizing SD-trees.

Unfortunately, all of this was not quite enough to make PPG strictly better than an unguided path tracer under simple illumination. In some scenes—for example directly lit outdoor environments—unidirectional path tracing sometimes outperformed our extended PPG by a small margin that is mostly caused by the 10-20% computational overhead of the SD-tree and our discarding of all but the last 4 iterations (around 6.25% of all samples). Nevertheless, our extensions significantly shrunk the gap between both approaches while improving PPG in difficult situations beyond what it was capable of before.

Additionally, our extensions are not inherently limited to PPG. With minor modification, they are also compatible with alternative guiding schemes [Vorba et al., 2014], other incident-radiance representations [Müller et al., 2018, Herholz et al., 2019, Vorba et al., 2014], and even different paradigms, such as guiding in primary-sample space [Müller et al., 2018, ?, ?] or path space [Simon et al., 2018].

We are excited about the ongoing adoption of path guiding in movie production and look forward to further progress that will be made in the field.

Acknowledgments

We thank Disney's Hyperion team for having been invaluable in integrating PPG into Hyperion and developing the presented extensions; in particular, we thank Laura Lediaev, Joseph Schutte, Peter Kutz, Ralf Habel, Yining Karl Li, and David Adler. We also thank thank Jan Novák for help with the SD-tree illustration and Brent Burley, Dan Teece, Fabrice Rousselle, Jan Novák, Joseph Schutte, and Sebastian Herholz for proofreading and helpful feedback. Lastly, we thank the following entities and people for providing scenes and models that appear in our figures: Benedikt ?, Ondřej Karlík (Swimming Pool), Jay-Artist (Country Kitchen), and SlykDrako (Bedroom).

11 Volumetric Zero-variance Based Path Guiding

Sebastian Herholz, *University of Tübingen*

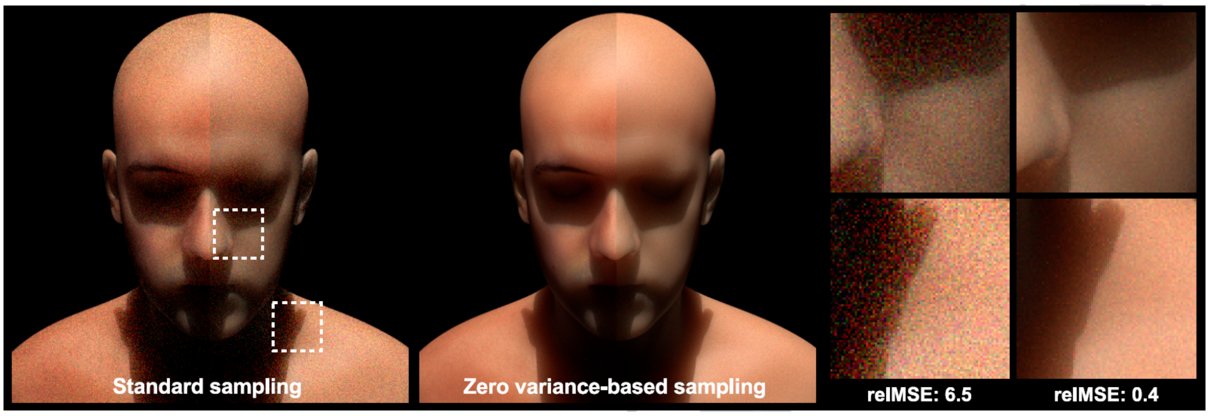


Figure 17: Path tracing of an optically dense medium (30 minutes), showing both a complete 'beauty' render *(S+V)*, and a render with only volumetric transport *(V)*. Building on an approximate adjoint solution of the incident and in-scattered radiance, our zero variance-based path construction forms near-optimal decisions for guided collision distance sampling, directional sampling, Russian roulette and path splitting. As such, our sampling methodology leads to significantly faster convergence compared to an unguided path tracer with standard transmittance-based collision and phase function sampling.

11.1 Introduction

Recent work on path guiding in volumes Herholz et al. [2019] applies strict rules given by zero-variance theory. We will demonstrate the importance of guiding all individual sampling decisions (directional, distance, Russian roulette or splitting) when aiming to reduce the variance of the MC estimator and will explain how parametric mixture models can be used to represent incident and in-scattered radiance needed to guide these decisions. Furthermore, we will discuss details one needs to consider when implementing path guiding in a production renderer such as *Weta Digital's Manuka*.

Volumes and why are they complicated? Introducing volume into scenes and supporting the solution of the correct volumetric light transport is a crucial part when modeling realistic scenes and effects. Volumetric light transport happens in many different scenarios such as clouds, fog, subsurface scattering and liquids. Unfortunately, integrating volumetric transfer into the rendering equation (RE) introduced by Kajiya [1986], which is solved by the rendering system, increases its complexity. This increased complexity can have a high impact in the efficiency of current rendering systems, especially unidirectional path tracer, that increase the time needed until a rendering reaches a converged state. Especially effects such as direct/indirect light shafts, dense high albedo media (e.g. clouds, skin) or volumetric caustics are effects, which are hard to render with a simple uninformed/guided path tracer. In the recent years many specialized techniques such as, *equi-angular sampling* (Kulla and Fajardo [2012]), *joint-importance sampling* (Georgiev et al. [2013]) or *dwivedi based zero-variance sampling* (Křivánek and d'Eon [2014] and Meng et al. [2016]), have been developed to optimize the solution of the volumetric rendering equation. Each of them is able to efficiently solve one specific case of the volumetric transport, but none of them, or even their combination is able to resolve to an optimal sampling procedure for the complete volumetric transport. Our volumetric path guiding approach attempts to overcome these limitations by approximating the optimal sampling for the complete volumetric transport. The additional challenge for generating paths in the present of volumes, is that they also need to solve the additional volumetric contribution L_m, which is added to the standard RE:

$$L(\mathbf{x}, \boldsymbol{\omega}) = \underbrace{T(\mathbf{x}, \mathbf{x}_s) L_\mathrm{o}(\mathbf{x}_s, \boldsymbol{\omega})}_{L_\mathrm{s}(\mathbf{x},\boldsymbol{\omega})} + \underbrace{\int_0^s T(\mathbf{x}, \mathbf{x}_d)\sigma_\mathrm{s}(\mathbf{x}_d) L_\mathrm{i}(\mathbf{x}_d, \boldsymbol{\omega}) \,\mathrm{d}d}_{L_\mathrm{m}(\mathbf{x},\boldsymbol{\omega})}, \tag{17}$$

where L_s describes the attenuated surface contribution L_s which is scattered from behind the volume. One source of the increased complexity of the VRE is the fact, that solving the volumetric contribution added the need of solving the integral of the in-scattered radiance L_i along the ray ($[0, 1]$). The other is the fact that the in-scattered radiance is again a volumetric transport quantity, that is evaluated by solving the spherical integral:

$$L_\mathrm{i}(\mathbf{x}_d, \boldsymbol{\omega}) = \int_S f(\mathbf{x}_d, \boldsymbol{\omega}, \boldsymbol{\omega}') L(\mathbf{x}_d, \boldsymbol{\omega}') \,\mathrm{d}\boldsymbol{\omega}'. \tag{18}$$

Compared to solving the surface based light transport, which only depends on making a directional decision and termination at each surface intersection. Solving the volumetric transport adds four additional decisions (see Figure 11.1), that need to be made to the path generation process of a path tracer:

1. The *scatter decision* determines, whether the path should explore the volumetric contribution L_m or the attenuated surface contribution L_s of the VRE.
2. If L_m is be explored the *distance decision* defines, where the next scattering event along the ray should be generated.
3. The subsequent *directional decisions* at the new scattering position inside the volume defines how the in-scattered radiance L_i is explored.
4. To determine whether the path inside a volume should be continued or not a path *termination decision* has to be made.

These four decisions are repeated recursively, until the generated path is terminated or leaves the scene. Since making these four decisions in an optimal way would require prior knowledge about the solution of the VRE itself, a common way is to make these decisions only based on the known local volume properties. For more details about the commonly used process to sample volumes in production we refer the reader to Novák et al. [2018] and Novák et al. [2018].

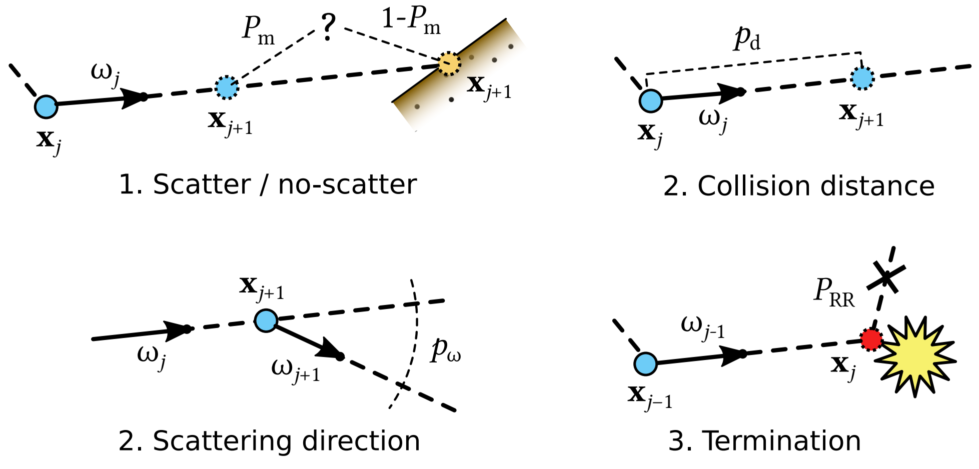


Figure 18: Visualization of the four different sampling decisions, which need to be made when generating a random path inside a volume.

Standard Sampling Techniques and Zero-Variance Theory

Using the zero-variance random walk theory as explained by Hoogenboom [2008] it is possible to derive the optimal PDFs and probabilities for making the previously introduced four decisions. It can be seen as a strict framework, to make importance sampling decisions for the complete integrand of the VRE. A path tracer, which makes all sampling decisions strictly based on this theory would lead to a zero-variance estimator, which only needs to evaluate one sample per pixel to generate a converged image. Unfortunately, these decisions rely on exact estimates of all volumetric transport quantities upfront, which would imply already having a solution for the complete volumetric light transport we currently want to solve. Especially the quantities for the *incident radiance* L and *in-scattered radiance* L_i are important to be known upfront and at each position and direction in the scene, to construct optimal sampling decisions. Since this information is usually not available upfront, common sampling strategy neglect the influence of the incident and in-scattered radiance and on samples based on the transmittance and on the scattering behavior of the phase function. Figuratively speaking one could say that these sampling decisions assume that both quantities L and L_i are constant across the whole scene, and would resolve in a zero-variance estimator, if and only if this assumption is true. Therefore, the variance of the commonly used volumetric path tracer is related to the derivation of actual volumetric light transport of a scene to the assumption of constant L and L_i. As a consequence any biasing/guiding of the sampling decisions in the direction of the optimal *zero-variance* ones will lead to a reduction in the variance of the random walk estimator. The goal of our guided volume path tracing approach is to use local approximations of these quantities to guide **all** four decisions in the direction of the optimal ones dictated by the zero-variance theory.

11.2 Volumetric Path Guiding

In recent years the concept of path guiding has proven to be a reliable method to reduce variance/error of a rendering system for surface based light transport. Our volumetric path guiding approach, extends this concept to also support guiding the path generation process in the present of participating media. Early approaches of volumetric path guiding (Bashford-Rogers et al. [2012] and Pegoraro et al. [2008]) only focus on guiding the directional sampling decision. Due to the lack of supporting guiding based on the product of incident radiance and the phase function, both methods are limited to only support guiding for near isotropic volumes. Instead of just applying guiding to the directional sampling decisions, as done when guiding surface based light transport, our approach considers guiding all four additional decisions, which needs to be made, when sampling a random path inside a volume. Furthermore, our guided decision are strictly related to the optimal zero-variance ones as described by Hoogenboom [2008], which includes directional guiding based on incident radiance and phase function product. In the following section we are going to show that guiding all four decision at once and that the strict relation to the optimal zero variance theory is crucial, when trying to reduce the variance/error of a volumetric path tracer.

Conceptually our volumetric guiding approach is based on a similar guiding structure as used by recent surface based guiding techniques (Vorba et al. [2014] and Müller et al. [2017] Section 10). Using a spatial sub-division structure, such as a BSP-tree, we are able to store local approximations of the relevant transport quantities used to guide our sampling decision. In case of the volumetric transport, these estimates are spherical representations of the incident and the in-scattered radiance L and L_i (Section 11.3). A visualization of this structure is shown in Figure 11.2. Using this guiding representation, we are able to derive the guided sampling decisions (Section 11.4). These guided decisions are based on the strict rules of the zero-variance theory and cover all four sampling steps which need to be considered, when generating a random path in the presents of a volume.

In the following sections we will first introduce the representation used by our approach to store the spherical distribution of the needed transport quantities L and L_i (Section 11.3). Then we will show how these representations can be used to build guided sampling strategies that fulfill the requirements of the zero-variance theory (Section 11.4.1, Section 11.4.2, Section 11.4.3). For each of these individual decisions we will show the effect on reducing the error/variance of the final rendering and how this reduction increases if they are combined. Since the crucial part of our guiding approach is the structure used to represent the guiding estimates for L and L_i, we will give detailed information about how such a structure can be trained/learn using different strategies, such as photon based pre-processing step or the online forward learning approach by Müller et al. [2017] (Section 10). We will also mention production relevant

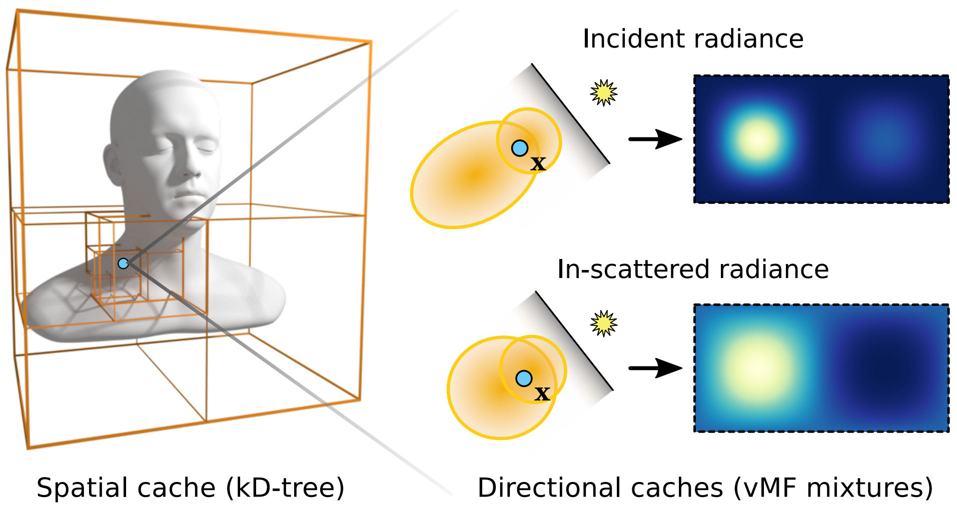


Figure 19: *Left:* Visualization of the (kD) tree for the spatial cache component. *Right:* A schematic and false-color depiction of the directional caches for the incident and in-scattered radiance represented via vMF mixtures (*orange*). The sample volume has a slightly forward-peaked phase function.

problems, which may arise, when integrating our approach in a production environment such as *Weta Digital's Manuka*(Section 11.5).

11.3 Volumetric Radiance Estimates

The key components of our volumetric guiding approach is the representation of the volumetric light transport quantities needed to make make guided sampling decisions based on zero-variance random-walk theory. Both the *incident radiance* L and the *in-scattered radiance* L_i are quantities, which can be defined as spherical functions for a given position x inside the volume. While L defines the incoming radiance arriving at x from any direction ω, L_i defines the amount of radiance, which is in-scattered into a direction ω at the position x. Based on the characteristics of the volumetric light transport in a scene L and L_i can vary strongly based on the position and direction in the scene/volume, which can lead to a large number of number of guiding distribution/caches, which needs to be stored to represent this characteristics correctly. Therefore, the challenge in finding a suitable representation is not only the efficiency of evaluating the needed quantity but also in the memory footprint needed to store one representation. A compact representation of a spherical function can be achieved via a mixture of weighted simplified spherical distributions such as von Mises Fisher (vMF) lobes. A vMF mixture model V is composed of a set of K weighted lobes v:

$$\mathrm{V}(\boldsymbol{\omega} \mid \Theta) = \sum_{i=1}^{K} \pi_i \cdot \mathrm{v}(\boldsymbol{\omega} \mid \boldsymbol{\mu}_i, \kappa_i). \tag{19}$$

The parameter set Θ contains the weights $\{\pi_1, ..., \pi_K\}$, the mean directions $\{\mu_1, ..., \mu_K\}$ and precisions $\{\kappa_1, ..., \kappa_K\}$ for each component ('lobe'). Each vMF lobe is a normalized spherical distribution, rotationally symmetric around its mean, with its spread being inversely proportional to κ. To represent the a spherical distributions at different locations $\mathbf{x}$ inside a medium, the vMF mixtures V can described by the spatially-varying set of parameters $\Theta(\mathbf{x})$. These sets can be stored in a BSP-tree structure similar as the one used by Müller et al. [2017]. An important features of the vMF mixtures is, that they inherit the features from the used vMF models, such as efficient sampling, analytical integration, convolution and product calculation. The evaluation and these inherited features can be efficiently implement for multiple components at once using vector instructions such as SEE (4) or AVX (8).

11.3.1 Incident Radiance Estimate

To represent the incident radiance estimate $\tilde{L}$ using vMF mixtures one needs to scale the evaluation of the normalized spherical distribution by the scalar irradiance (i.e. fluence) at x:

$$\tilde{L}(\mathbf{x}, \boldsymbol{\omega}) = \Phi(\mathbf{x}) \cdot \mathrm{V}_L(\boldsymbol{\omega} \mid \Theta_L(\mathbf{x})). \tag{20}$$

Fluence is given by $\Phi(\mathbf{x}) = \int_S L(\mathbf{x}, \boldsymbol{\omega}') \, \mathrm{d}\boldsymbol{\omega}'$. Based on the source of the data used to fit the vMF mixture for the incident radiance (Section 11.5.1) its value can be either determined directly from incident radiance samples or via photon density estimation.

11.3.2 In-Scattered Radiance Estimate

Further exploration of the relationship between the incident radiance L and the in-scattered radiance L_i reveals, that L_i is given by convolving the incident radiance spherically with the phase function (Equation 18). If the incident radiance is estimated by the vMF mixture V_L the estimate for the in-scattered radiance $\tilde{L}_\mathrm{i}$ resolve to:

$$\tilde{L}_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = \Phi(\mathbf{x}) \cdot \int_S f(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}') \, \mathrm{V}_L(\boldsymbol{\omega}' | \Theta_L(\mathbf{x})) \, \mathrm{d}\boldsymbol{\omega}'. \tag{21}$$

Since phase functions are typically rotationally invariant they can be represented with one or more vMF lobes. The convolution of two vMF lobes results in another vMF lobe, which can be calculated analytically (see the Appendix of Herholz et al. [2019]), providing a highly efficient way to obtain the integrated in-scattered radiance estimate:

$$\begin{aligned} \tilde{L}_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) &= \Phi(\mathbf{x}) \cdot \mathrm{V}_{L_\mathrm{i}}(\boldsymbol{\omega} | \Theta_{L_\mathrm{i}}), \\ \mathrm{V}_{L_\mathrm{i}}(\boldsymbol{\omega}) &= (\mathrm{V}_f * \mathrm{V}_L)(\boldsymbol{\omega}). \end{aligned} \tag{22}$$

The convolution of a mixture with K components for the incident radiance and a mixture with L components, representing the phase functions, leads to a mixture for the in-scattered radiance with $K \cdot L$ components. To avoid additional storage for this mixture and to support the usage of multiple differing phase function it is possible to adjust the evaluation process of the vMF mixture to support the convolution of the mixture on-the-fly with one vMF lobe. This adjustment leads to L calls of the adjusted evaluation function of the incident radiance vMF mixture with K components.

Examples of our estimates compared to the ground-truth measurements are given in Figure 11.3.2.

11.4 Guided Sampling Decisions

Using the vMF representation of the incident and in-scattered radiance we are now able to derive multiple guided sampling decisions to cover the four decisions needed to generate a random path in the presence of a volume. In the following section we will introduce the zero-variance PDFs/probabilities for all guided decisions and show how one can guide the path generation process based on approximations of the optimal ones. Since the all our guided decisions try to approximate the complete optimal PDFs, it is worth noting that, in the hypothetical case, that the used guiding estimates for the incident and in-scattered radiance and the vMF mixture representation of the phase would be exact, our guided sampling would converge to the optimal zero-variance one.

11.4.1 Guided Product Distance Sampling

The sampling of the next scattering location along a path segment includes the two different sampling decisions. The first one decides, if the scattering event should explore the volume contribution L_m or the surface contribution L_s from behind the volume. If the decision is made to explore the volume, the distance along the path segment inside the volumes has to be sampled. Fortunately these two decision can be merged into a single distance sampling decision, which either samples a distance to a point x inside

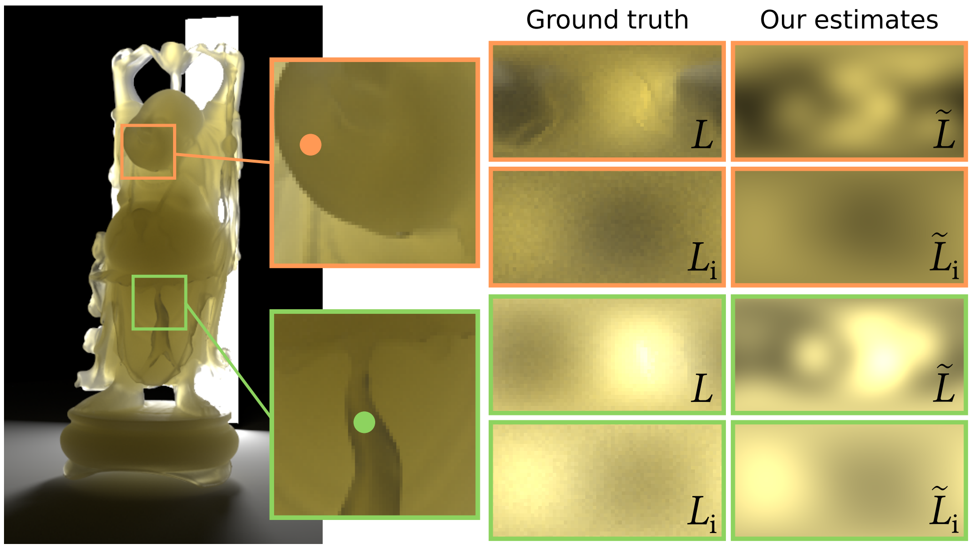


Figure 20: Our adjoint solution estimates for the volumetric radiance quantities L and L_i for two positions below the surface of the Buddha statue. The center column shows the ground-truth spherical function of L and L_i evaluated using path tracing with 100k samples per pixel. The right column shows our estimates $\tilde{L}$ and $\tilde{L}_\mathrm{i}$ fitted from 1k photons. Note that since L_i is an integrated quantity, its estimate is much more reliable than that of L.

the volume, or the sampled distance passes the volume and resolves to the distance to the point x_s on the nearest surface behind the volume. The zero-variance PDF for this joint distance sampling is:

$$p_\mathrm{d}^{zv}(d \mid \mathbf{x}, \boldsymbol{\omega}) = \frac{T(\mathbf{x}, \mathbf{x}_d) \cdot \sigma_\mathrm{s}(\mathbf{x}_d) \cdot L_\mathrm{i}(\mathbf{x}_d, \boldsymbol{\omega})}{L(\mathbf{x}, \boldsymbol{\omega})}, \tag{23}$$

where, the denominator is the sum of the attenuated surface contribution and the volumetric radiance contribution ($L_\mathrm{s} + L_\mathrm{m} = L$).

Incremental Guided Product Distance Sampling

To guide the combined distance sampling decision in the sense of the optimal zero-variance one as defined in Equation 23, one needs to consider not only the in-scattered radiance L_i but also the incident radiance L, that arrives at the beginning of the path segment at x. Using our guiding representation (Section 11.3) we are able to estimate these quantities ($\tilde{L}$ and $\tilde{L}_\mathrm{i}$) and construct a distance sampling methods, which is able to approximate the optimal one from Equation 23. Our proposed sampling method is inspired by the simple procedure of tabulating the path segment, that intersects the volume by separating the distance into multiple bins ($d_1, ..., d_N$). The size of these bins can be either be defined by the sizes of the spatial structure of our guiding estimates or regular tracking based on the local volume properties. Under the assumption that the in-scattered radiance and the volume properties (e.g. σ_t and σ_s) are constant inside one bin it is possible to evaluate the contribution of each bin to the incident radiance arriving at x. Using these contributions for each bin and by evaluating the incident radiance at the end of the volume intersection it is possible to build a discrete CDF and sample a bin according to it. The probability of sampling one specific bin d_i is then based on Equation 23.

Unfortunately this simple approach has one major drawback, which comes from the fact that the complete volume has to be traversed and that the contribution of **all** bins has to be evaluated. This is necessary to calculate the normalization factor of the CDF, which is needed to derive the probability of each bin. This can lead to massive computational overhead, especially in production when dealing with large scenes or dense media, where most of the transport happens close to the entry point of the volume (e.g. subsurface scattering of skin). To overcome this limitation we developed an incremental sampling approach, which is able to use the local estimates $\tilde{L}$ and $\tilde{L}_\mathrm{i}$ to make subsequent decision, if a scattering

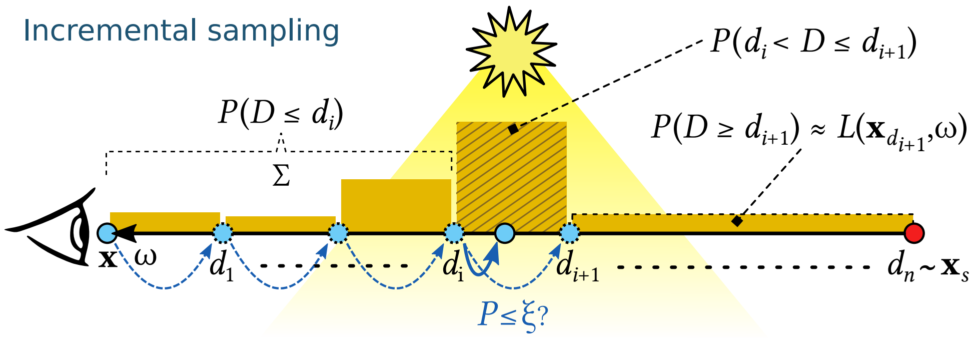


Figure 21: Our incremental technique (Algorithm 4) steps along the ray, proposing candidate bins according to the current PDF estimate and the estimated probability mass along the remainder of the ray without explicitly sampling it. This approach is especially beneficial for optically dense media, since on average fewer steps are taken due to the rapidly decreasing transmittance function with regard to increasing d.

event should be generated inside the current bin d_i or not. This incremental nature of the algorithm bounds the needed memory access and evaluations to number of bins, which were passed before the scattering event is created. The ability to build this incremental approach is based on the fact, that the probability to scatter in a bin that starts at d_i and ends at d_{i+1} mainly depends on the ratio between the in-scattered radiance and the incident radiance arriving at the distance d_i:

$$P_i(D \leq d_{i+1}) \approx \frac{1 - T(d_i, d_{i+1})}{\sigma_t(d_i)} \cdot \frac{\sigma_s(d_i) \cdot \tilde{L}_i(d_i)}{\tilde{L}(d_i, \boldsymbol{\omega})}. \tag{24}$$

This probability can be derived, when setting x in Equation 23 to the distance d_i (since we already reach this point without generating a scattering event) and then integrate Equation 23 from d_i to d_{i+1}. Since it is assumed that $\sigma_s(x)$ and $L_i(x)$ are constant inside the bin only the transmittance needs to be integrated, which resolves to the first factor of Equation 24.

The probability of not generating a scattering event in the previous steps, until we reach the bin starting at d_i, resolves to:

$$P(d_i < D) = \prod_{j=0}^{i-1} 1 - P_j(D \leq d_{j+1}), \tag{25}$$

and can be incrementally updated while stepping over the preceding bins. To sample the final position inside a selected bin, we use a similar approach as presented by Kulla and Fajardo [2012], which enables sampling based on the transmittance inside a given boundary $[d_i, d_{i+1}]$.

The performance of this proposed distance-sampling method mainly depends on the quality of the estimates of L and L_i and the resolution of the spatial cache structure. In the hypothetical case, that these estimates are correct and the bin sizes are small enough, so that the assumption of a constant L_i and σ_s holds, the method converges to optimal zero-variance sampling decision from Equation 23. To compensate to for sub-optimal decisions, which caused by inaccurate estimates we combine our incremental distance with commonly used transmittance distance sampling using the one-sample MIS model (Veach [1997])and the selection probability $\alpha_d = 0.5$. The final PDF for sampling a distance d using these combined methods is therefore:

$$p_d(d_{i+1}|r_i) = \alpha_d \cdot p_d^{\tilde{zv}}(d_{i+1}|r_i) + (1 - \alpha_d) \cdot p_d^{std}(d_{i+1}|r_i), \tag{26}$$

where $p_d^{\tilde{zv}}$ represents our approximated version of the optimal zero-variance PDF. The pseudo code of our incremental guided distance sampling is listed in Algorithm 4. For more information and further details about the derivation and a method to increase the stability of the sampling, we refer the reader to the related section in the original paper of Herholz et al. [2019].

Some examples of the effects of our guided distance-sampling approach is shown in Figure 25 and Figure 25.

Algorithm 4: Algortihm for incremental guided distance sampling.

```
function guidedDistanceSampling($\mathbf{x}$, $\boldsymbol{\omega}$, $dMax$)
    // $\mathbf{x}$ ...starting position inside or at the beginning of the medium
    // $\boldsymbol{\omega}$ ...direction in which the walk continues
    // $dMax$ ...distance to the next surface intersection
    $d_i := 0$
    $pdf := 1$
    $scatter := False$
    while not $scatter$ and $d_i < dMax$ do
        $\mathbf{x}_i := \mathbf{x} - d_i\boldsymbol{\omega}$
        $[\tilde{L}, \tilde{L}_\mathrm{i}]$ := lookUpEstimates($\mathbf{x}_i$, $\boldsymbol{\omega}$)          // Sec. ??
        $[\sigma_\mathrm{t}, \sigma_\mathrm{s}]$ := getMediumCoefficients($\mathbf{x}_i$)
        $d_{i+1}$ := getDistanceToNextBin($\mathbf{x}_i$, $\boldsymbol{\omega}$)
        $P$ := calcBinProbability($\tilde{L}, \tilde{L}_\mathrm{i}, \sigma_\mathrm{s}, \sigma_\mathrm{t}, d_i, d_{i+1}$)          // Eq. 24
        $\xi$ := getRandomValue( )
        if $\xi \le P$ then
            // current bin selected
            $pdf *= P$
            $[d, pdfBin]$ := sampleDistanceInBin($d_i, d_{i+1}, \sigma_\mathrm{t}$)
            $pdf *= pdfBin$
            $scatter := True$
        else
            // continue tracking
            $pdf *= (1 - P)$          // Eq. 25
            $d_i := d_{i+1}$
    return $[scatter, d, pdf]$
```

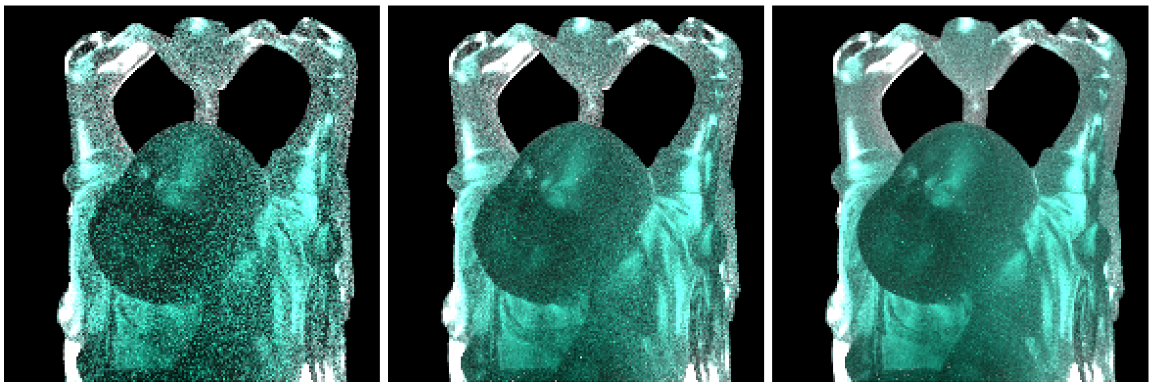


Figure 22: Comparison of different sampling methods (256)spp: standard distance and directional sampling (**left**), our guided distance sampling (**middle**) and our guided distance and directional sampling (**right**). In the scene a dense backward scattering volume is illuminated by a small light source from behind. While the standard distance sampling approach generates early scattering events inside the dense volume, our guided product distance samples avoids these early scattering events and passes through the dense media towards the light source. Adding directional product guiding further reduces the error/variance by guiding the scattering events in the direction of the light source and not back to the camera.

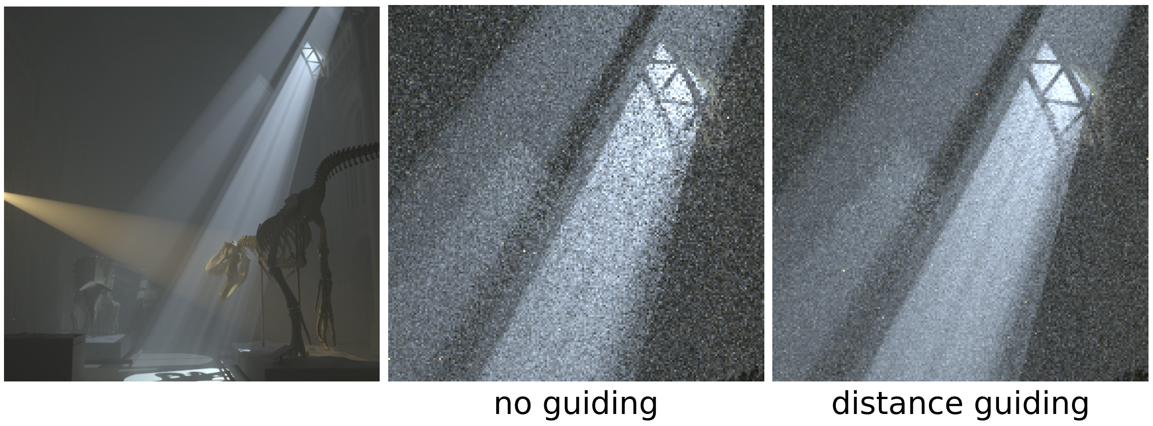


Figure 23: Comparing standard distance sampling (**middle**) against our guided distance sampling (**right**) in a scene with a thin volume illuminated by spotlight sources (1024spp). While the standard transmittance based approach has a low probability in generating scatting events inside the light shafts, our guided product distance sampling approach generates more scattering events inside these regions. Since our approach is also applied for multiple scattering events, this more optimal decisions also reduce the variance in the multiple scattering component of the volumetric light transport.

11.4.2 Guided Product Directional Sampling

Guiding the directional sampling decision plays an important role in reducing the error of a volumetric path tracer. After the decision is made, where inside the medium a scattering event is generated, it is important that the in-scattered radiance at this point is explored in the right direction. Commonly the directional sampling is governed by the phase function, yet the phase function is frequently a poor approximation of the actual propagation of light. As an example one can consider a dense medium with an highly backward scattering phase function. This medium is mainly lit from behind the the medium, towards the camera, by a strong light source. Phase function based sampling would mainly sample directions that point away from the light source, which leads to a high variance estimates. Another example is a scene with isotropic fog with many light shafts. In this example the advantage of being able to generate a sample inside the light shaft, using our guided product distance sampling, is worthless, when the followed directional sampling decision does not explore the direction towards the source of the light shaft.

Our approach of guiding the directional sampling decisions inside a volume follows from trying to optimally importance sample the in-scattered radiance integral by sampling ad direction based on the product of the incident radiance and the phase function. This sampling behavior directly corresponds to the optimal zero-variance PDF:

$$p_{\omega}^{zv}(\omega_{i+1}|\omega_i, x_{i+1}) = \frac{f(\omega_i, \omega_{i+1})L(x_{i+1}, \omega_{i+1})}{L_{\mathrm{i}}(x_{i+1}, \omega_i)}. \tag{27}$$

To achieve such a sampling we utilize the fact, that the product of two vMF lobes form one vMF lobe, which can be constructed efficiently in closed form. This, in turn, allows us to also construct products of vMF mixtures in closed form. We can therefore use our vMF representations of the incident radiance and the phase function of the volume to generate a product vMF mixture. The product mixture $\mathrm{V}_{\otimes}$ is calculated based on the mixture V_L of the incident radiance and the mixture V_f of the phase function:

$$\mathrm{V}_f(\boldsymbol{\omega})\mathrm{V}_L(\boldsymbol{\omega}) = (\mathrm{V}_f \otimes \mathrm{V}_L)(\boldsymbol{\omega}) = \mathrm{V}_{\otimes}(\boldsymbol{\omega}). \tag{28}$$

A new random scattering direction $\boldsymbol{\omega}$ is obtained by importance-sampling this product mixture following the stable procedure presented by Jakob [2012]. This approach can be seen as a direction extension of the product guiding method for surfaces introduced by Herholz et al. [2016]. In Contrast to the Gaussian mixtures used by Herholz et al. [2016] we use vMF models, which better correspond to the spherical characteristics of the volumetric transport quantities. For the exact formulas of how the product mixture is calculated and how to deal with vMF lobes that represent forward or backward scattering features of a phase function we refer the reader to the Appendix of the original paper by Herholz et al. [2019].

Similar to our guided distance sampling approach we use multiple importance sampling (MIS) between our product mixture and phase-function sampling to account for inaccuracies in the guiding estimates. The combined PDF for sampling a new direction in the volume including our approximated zero-variance one $p_\omega^{\tilde{zv}}$ and the PDF for standard sampling based on the phase function is:

$$p_\omega(\omega_{i+1}|x_{i+1},\omega_i) = \alpha_\omega \cdot p_\omega^{\tilde{zv}}(\omega_{i+1}|x_{i+1},\omega_i) + (1-\alpha_\omega) \cdot p_\omega^{std}(\omega_{i+1}|x_{i+1},\omega_i). \tag{29}$$

The probability of sampling based on our product mixture is defined by the balance heuristic weight α_ω. In practice we found that a conservative value of $\alpha_\omega = 0.5$ leads to be reliable enough to compensate for potential inaccuracies in the fits, but still leads to significant decreases in the error of the final renderings.

An example of the effect of our guided directional-sampling approach is shown in Figure 11.4.2. In Figure 11.4.2 we demonstrate the importance of being able to guide according the product of the incident radiance and the phase function. Especially in dense media and phase functions with higher mean cosine values, the neglection of the product can lead to even worse sampling behavior as when using no directional guiding at all.

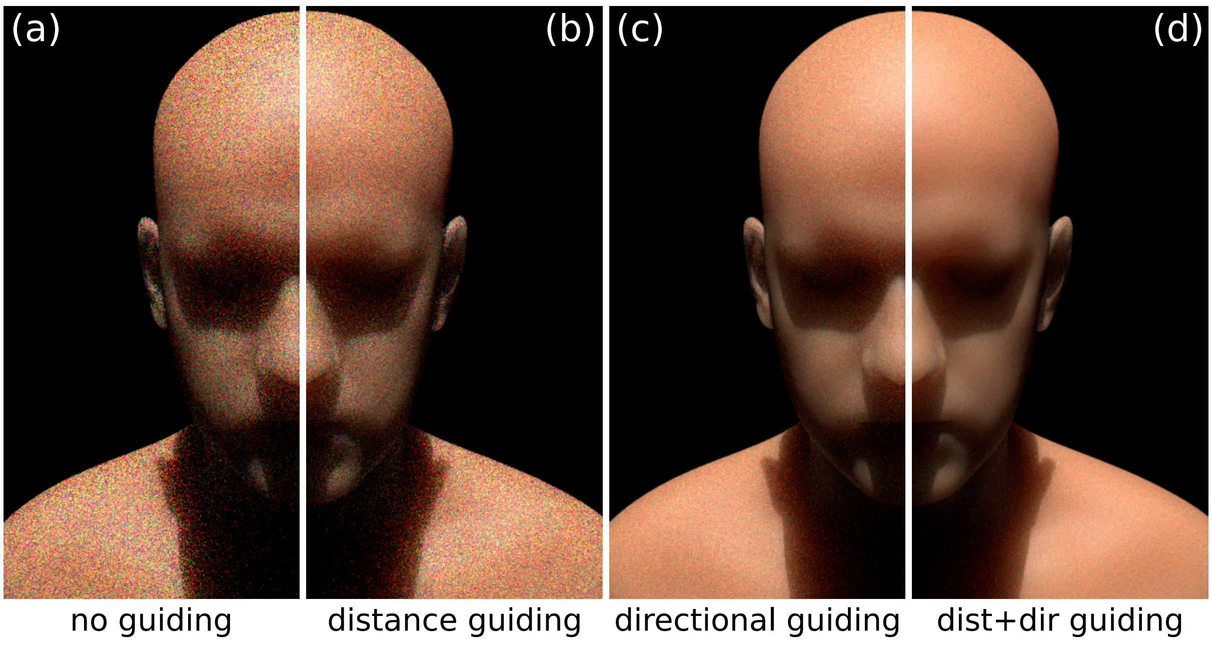


Figure 24: Comparing different volume sampling methods (256spp): standard distance + directional sampling (**a**), only guided distance sampling (**b**), only guided directional sampling (**c**) and the combination of guided distance and directional sampling (**d**). While in this setup, guiding the distance sampling decision only has a minor impact on the variance/noise of the render, guiding the directional sampling decision has a large impact on reducing the error/variance. Nevertheless the best result can be achieved, when both guiding methods are combined.

11.4.3 Guided Russian Roulette and Splitting

Russian roulette (RR) and splitting are two important tools, which can be used to influence the performance of a rendering system. While the target of RR is to increase the efficiency of the render, to be able to evaluate more samples at a given time budged, splitting mainly focuses on reducing the error generated by the render, by reducing the variance of Monte-Carlo process. Nevertheless setting up these two techniques can be quite complicated and non-optimal settings can either lead to an increase in variance (e.g.

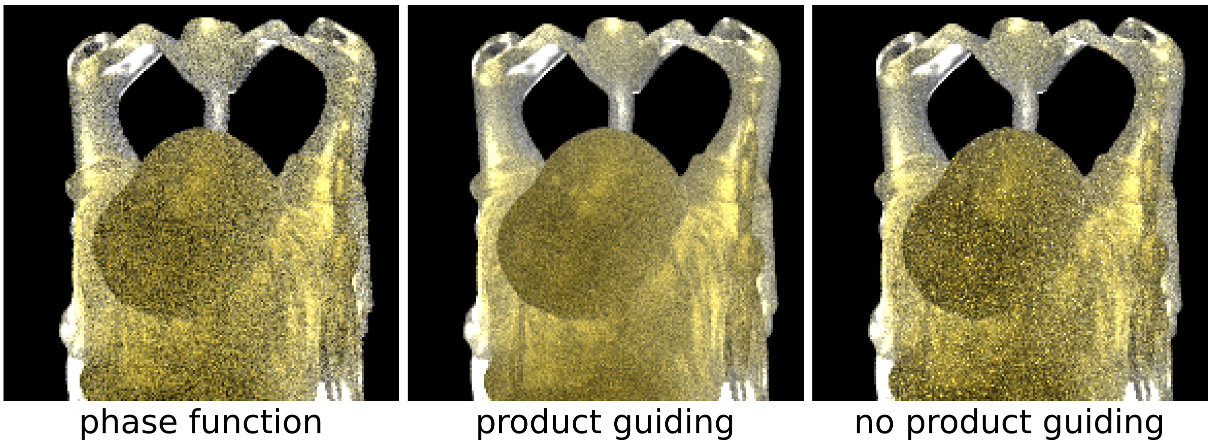


Figure 25: Comparing different directional sampling methods (256spp): standard based on the phase function (**left**), our guided directional product (**middle**) and a guided version just based on MIS between the phase function and the incident radiance (**right**). Especially in dense volumes with high mean cosine, being able to sample according the product becomes important.

early path termination using RR) or can negatively impact the efficiency of the renderer (e.g. to aggressive splitting). Another challenge is that the optimal settings for RR and splitting are often not global and can be scene or even dependent on the local position inside a scene. This is also true when dealing with volumetric light transport and therefore guiding these decisions plays an important role in optimizing a rendering system.

Russian Roulette

The concept of Russian roulette is to increase the efficiency of an estimator by terminating paths, which have a potentially low contribution to the final estimate (pixel value). To avoid the costly evaluation of these low contributing path, they are terminated stochastically based on a termination probability P_{RR}. The equivalent survival probability for a path is $q = 1 - P_{RR}$. The goal of this probability is to estimate the future/potential contribution of a path when its construction and evaluation is continued. Figuratively speaking the main question to answer is:"Will the path hit a light source in the future? And if so, will the contribution of this light source propagated through the path high enough to effect the estimate?" If this decision is optimal, RR would terminate paths, which got lost in space, by non-optimal former sampling decisions, and continue paths which are on their way to the light sources that mainly contribute to the final pixel value. This optimal termination keeps the additional introduced variance minimal, so that the increased efficiency leads to an overall lower variance at an equal time budget. Usually this decision is mainly based on the current throughput of the path or on the local scattering albedo. The intuition behind this heuristic is, that a path with low throughput or scattering albedo will also result in a low contribution to the final estimate, even if it will hit a light source in the future. While this assumption often holds, it can also lead to rather sub-optimal decision, which lead to early path terminations causing a large increase in the variance of the estimator (e.g. causing fire fly noise). Examples for this invalidation are scenes, where the contribution mainly comes from long path lengths or subsurface scattering with dense media, where multiple bounces inside the object occur before the path exits the object and then reaches a light source. If in these cases the paths are terminated before they hit the light sources the probability of even generating any valid path, which contributes to the pixel value is low, which leads to a high variance.

To avoid these problems the optimal survival probability can be defined as:

$$q = \frac{E[R]}{I}, \tag{30}$$

where $E[R]$ is the expected contribution of the path, when its evaluation is continued, and I is the solution of the final estimate (e.g. pixel value). The goal of guided RR is to estimate this optimal probability by using approximations of the future path contribution $E[R]$ and the finial pixel value I. There are multiple ways to approximate the value for the final pixel estimate I_{pix}. For example, in a progressive rendering framework the value can be estimates using a denoised version of the already rendered progression.

Another method which is used by Vorba et al. [2014] and Herholz et al. [2016] estimates I_{pix} by directly querying the guiding caches. For each pixel a primary ray is traced until the first diffuse intersection, while all light transport quantities (e.g. L_o, L_i) are queried from the caches along the way. Figure 11.4.3 gives an example of such an estimate, which is achieved by ray marching through the volume and integrating the in-scattered radiance estimates $\tilde{L}_i$. The estimation of the path contribution depends on the

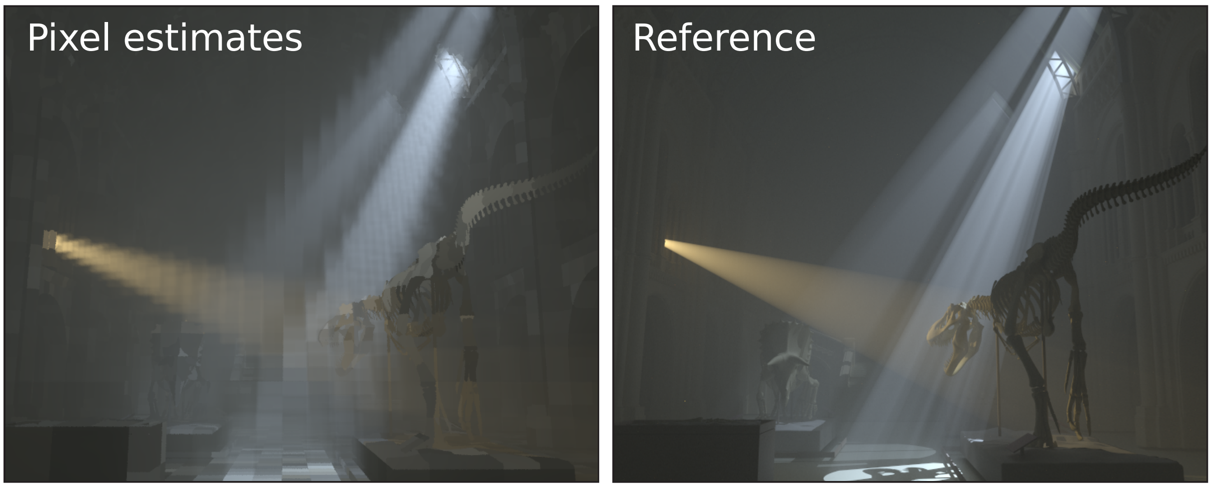


Figure 26: Comparison of the pixel estimates I_{pix} (generated by accumulating the in-scattered radiance from our estimates) to the ground-truth solution. Through the use of a *weight window* Vorba and Křivánek [2016] even this rough estimate is sufficient to guide our guided RR and splitting.

type of RR which is performed. When dealing with a volumetric path generation process it is possible to terminate the path in two different stages leading to two different RR approaches, which are based on two different probabilities. The first *directional RR* termination is done after a distance sampling step,

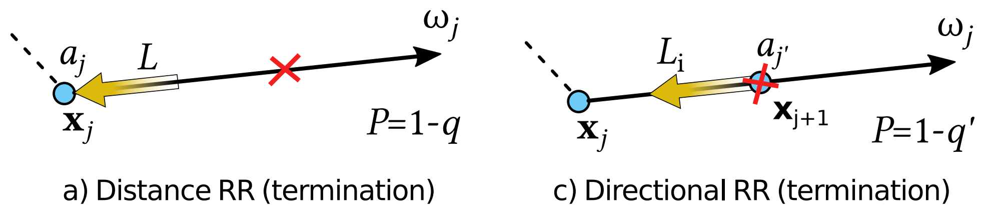


Figure 27: Volumetric guided RR. A volume RR decision (marked red) can either terminate the sampling of a new distance (**left**) or the sampling of a new direction (**right**).

which leads to a termination of the directional sampling step (Figure 11.4.3 right). The second *distance RR* decision can be made after a directional sampling step, which leads to a termination of the subsequent distance sampling decision (Figure 11.4.3 left). The survival probabilities q_{dir} and q_{dist} for these two RR decisions are based on the two different estimates for the future path contributions $E[R]_{\text{dir}}$ and $E[R]_{\text{dist}}$. Using the estimate $\tilde{L}$ and $\tilde{L}_i$ from our guiding caches, these quantities can be estimated in combination with the current path throughput $a(R)$. The final survival probabilities for the *directional* and *distance* RR are:

$$q_{\text{dir}}(R) = \frac{E_{\text{dir}}[R]}{I_{\text{pix}}[R]} \quad \text{and} \quad q_{\text{dist}}(R) = \frac{E_{\text{dist}}[R]}{I_{\text{pix}}[R]}. \tag{31}$$

Detailed information about how to estimate the expected contributions $E[R]_{\text{dir}}$ and $E[R]_{\text{dist}}$ and their derivation can be found in the Section 6.3 of the original paper by Herholz et al. [2019]. To account for inaccuracies in the estimation of the path contributions and the pixel estimate I_{pix} the To account for inaccuracies in our estimates of the volumetric light transport quantities used to calculate the expected contributions and for inaccuracies in the pixel estimate we use the same *weight window* approach to determine q as Vorba and Křivánek [2016].

Splitting

The goal of splitting is to reduce the variance of an estimator by further exploring regions, where the primary estimator used to explore a part/quantity of the light transport equation has a high change to introduce a large error the final pixel value. This variance reduction is achieved by splitting the path at the current state its generation process and continue exploring the specific transport quantity (e.g. L or L_{i}) with K individual random paths.

If this splitting factor is derived optimally it leads to a major variance reduction of the final estimator, since the error introduced by each subsequent primary estimator is minimized. On the other hand, too aggressive/large splitting factors lead to a large number of additional path, which have to be evaluated by the render. If these additional path do not lead to a variance reduction at the subsequent estimator, the efficiency of the rendering system is decreased.

Guided splitting minimize the error of final pixel value, caused by paths that are sampled with a strategy that underestimated the actual contribution of the path. To identify these paths guided splitting uses the same survival probability used for the guided RR decisions (see Equation 30). In the case that a path is generated with a lower probability then its expected contribution the ratio of this contribution resolves to be greater one. To avoid that the error of the sample caused by the sub-optimal decisions used to generate the current path does not further increase, the evaluation of the local quantity of the light transport equation (e.g. L or L_{i}) is explored with more samples.

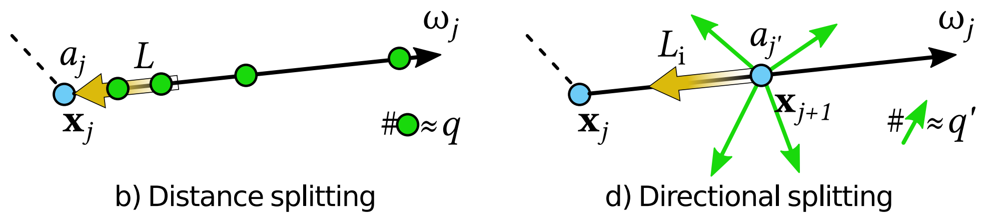


Figure 28: Volumetric guided splitting. A volume split (marked green) can generate both new sampled distances (**left**) and directions (**right**).

Similar to RR guided splitting can also be applied at two different stages of the random path generation process. The *directional splitting* is the opposite of *directional RR* and splits a path at a the position x_{i+1} into multiple sub-paths. This splitting leads to an average of $K = q$ individual estimates of the in-scattered radiance at x_{i+1}.

The *distance splitting* is the counterpart to *distance RR* and splits paths by generating multiple scattering events along the current ray segment r_i inside the volume. Its intention is to reduce the error of the estimate of the incident radiance arriving at x_i from the direction ω_i.

As mentioned by Vorba and Křivánek [2016] guided splitting can only act as a post variance reduction technique. Since it is apply after a sub-optimal sampling decision is made, it can not remove the variance introduced by a former decision afterwards. It only can bound the variance of the final estimator by trying to minimize the error introduced by all subsequent sampling decisions.

An example of the effects of guided RR and splitting is presented in Figure 11.4.3.

11.5 Details on the integration into a rendering system

The previous sections have shown the potential of path guiding for volumetric light transport and how it can be an efficient way to reduce the variance of a random walk path tracer. While the shown results focused on an implementation with a unidirectional path tracer Vorba et al. [2014] has shown that it is

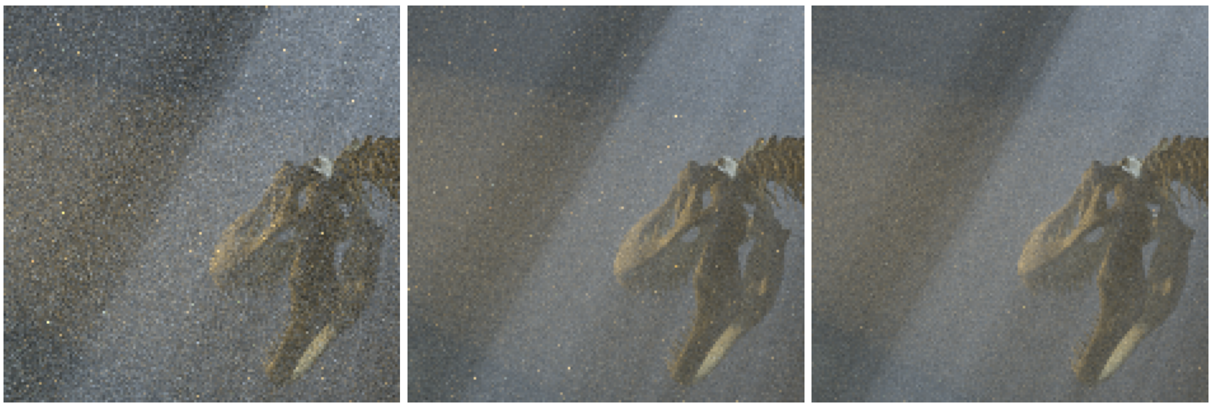


Figure 29: Comparing equal time renderings (45min) of a guided path tracer using: no RR or splitting (**left**), guided RR (**middle**) and guided RR and guided slitting (**right**). Guided RR reduces the error/noise of the final rendering by increasing the number of samples being evaluated at equal time. Splitting leads to an addition decrease of the error/noise by reducing the variance of the path tracer by splitting path with a potential high error/variance.

strait forward to also apply path guiding to bidirectional methods and that these also benefit a lot from it. Most of our presented methods to guide each of the sampling decisions are rather simple (e.g. directional guiding, guided RR) and an integration into a common or production rendering system would not be complicated. Nevertheless, the effectiveness and stability of the complete systems rises or falls with the quality an reliability of the vMF mixture fits for the incident radiance estimates and the vMF mixture representations for the phase functions of the used volumes. If these estimates and representations are not reliable enough the effect of volumetric path guiding vanishes, or can even lead to results with higher variance, compared to traditional sampling. On the other hand, since all our guided sampling decisions try to approximate the complete optimal PDFs based on the zero variance theory the more precise the estimates are the stronger will be the effect of the variance reduction in the final renderer. Compared to the current path guiding techniques for surfaces, where the potential variance reduction is not only based on the quality of the incident radiance estimates but also on how well the weighted sum of the PDFs of the BRDF and the incident radiance can approximate their product.

In the following section we are going to focus on aspects of how to estimate the vMF mixtures and what the necessary, important and practical considerations are, when integrating the system in a production oriented environment such as *Weta Digital's Manuka* or a scientific renderer such as *Mitusba* (Jakob [2010]).

11.5.1 Training/Fitting Incident Radiance Distribution

The following section will take a closer look on how the guiding caches (spatial and directional) can be trained using a data generated by a rendering system. Two different ways to generate the training data have been experimented on. The first method is the one presented in the paper by Herholz et al. [2019] and is based on a pre-processing step using a photon tracer. As a given number of photons is traced throughout the scene and then used to train the incident radiance field. The second method is based on the forward learning approach introduced by Müller et al. [2017] and which is explained in detail in Section 10. The basic idea is to gather data after each progression (sample iteration per frame) and update the guiding cache after a given amount of progressions (e.g. 2, 4, 8, ...).

Since there is only a minor difference in the overall procedure of build the guiding structure from these different methods, both methods are considered in the following paragraphs. At the end of this sections the pros and cons of both methods are compared and what needs to be considered when applying them a production environment.

Photon vs Forward Learning of vMF mixtures

To fit the vMF mixture for the incident radiance estimates, data from both a photon traced pre-processing step or radiance samples of the forward learning approach can be used. Both methods have their pros and cons. While the photon based pre-processing step distributes a fixed number of photon according to the characteristics of the light transport in the scene, the forward learning approach bypasses the need of any pre-processing time and gives the user a more or less interactive feedback. The quality of the samples thereby increases by every training iteration, since they benefit form use of path guiding. Usually the number of samples collected by the forward learning approach in each training step doubles with each iteration. Unfortunately, the current implementation of our EM fitting algorithm for the vMF mixtures requires to keep all these samples in memory when updating the estimates for the incident radiance distribution. This can quickly lead to a large additional memory overhead by the time the number of samples per iteration increases. On the other hand we observed that the vMF mixture are able to quickly fit reliable estimates even after a small number of training iterations and can do this even with a small amount of samples. Therefore, a way to compensate for the additional memory consumption is to update the vMF in smaller update cycles (e.g. every 32spp), or when collected samples reach fixed memory bound. Another, yet not investigated approach, would be the use of an online-EM algorithm similar to the one used by Vorba et al. [2014]. Since, we haven't found the optimal solution yet, and having a simple and reliable solution to learn vMF mixtures, by using the forward learning approach would be preferred solution, we see this as an interesting and important area of future work.

Training Data

The data used to fit the guiding distributions can either be generated in a pre-processing step via photon mapping (see. Herholz et al. [2019]), or generated on the fly by the renderer using a forward based learning approach, that generates samples based on the path traced during the current render progression (see. Müller et al. [2017]). As a result both methods starts with a set of samples that are distributed across the scene. The number of samples may vary (e.g. $5M$, $50M$ or even upto $250M$) but the main data, which needs to be stored per sample is quite similar:

Position	of the photon or incident radiance sample in the scene. This position is usually stored in world space coordinates.
Direction	from which the photon arrived at the sample position or the direction in which the random path went to generate the incident radiance sample. This direction in usually also stored in world space.
Energy	mainly depends on the type of samples. For photons this energy contains the power contributed by photon sample, while for radiance samples the energy is the actual incident radiance estimate from the direction of the sample.

In addition it is also useful to store the PDF of the sampled direction (in the case of forward-learning) and the throughput of the rest of the path until it reaches the emitting light source.

The main difference between photons and radiance samples lies in how they are representing the incident radiance distribution. For photons the radiance distributions depends on the power and the density of photons in each direction. The spherical distribution of the samples therefore also plays a role in estimating the incident radiance distribution. For incident radiance samples the important quantity is the estimated incident radiance of the sample direction. The role of the distribution of the samples mainly ensures the spherical domain of the incident radiance field and its major features are explored. These differences need to be considered, when fitting the distribution for the incident radiance. Fortunately for fitting our vMF mixtures this only leads to a minor of the fitting algorithm (see Section 11.5.1). The following equation shows an example of how the energy of a radiance sample at the jth scattering event of a random path R can be calculated:

$$\tilde{L}(x_j,\omega_j) = \underbrace{\prod_{l=j+1}^{M-1} a_l(r_l)}_{\text{throughput}} \cdot \underbrace{L_e(\mathbf{x}_M, \boldsymbol{\omega}_{M-1})}_{\text{emitted radiance}} . \tag{32}$$

For simplicity we only considered the case without next-event estimation (NEE).

The appearance of high variance sample (aka. fire flies) can have a high influence in the shape of the fitted incident radiance distribution. These high variance samples usually give a hint that the contribution from one direction has an higher importance then expected by the used sampling procedure. While for the forward learning approach by Müller et al. [2017], these high variance samples are used to explore the directional regions after the next training iteration in more detail, it is still important to distinguish between two types of fire flies. The first one is caused by paths, which were generated with a low probability and ended up hitting a bright light source. Since the contribution form the path was underestimated by the sampling procedure the already high energy from the light source gets boosted by the low sampling PDF and generates the firefly. These kind of high variance samples are exactly the ones, that identify directional regions, that are currently under-sampled and therefore an important information for the forward-learning approach.

The second type of fireflies occurs, if a path was sampled with a series of **extremely** low probability sampling decisions (e.g. low RR survival probabilities). Even when the light source at the end of the paths only emits a very low amount of energy the subsequent low probability decisions from the path generation process raises the contribution of the light source so much that a firefly evolves. These types of high variance samples are the ones that are undesirable when trying to estimate the incident radiance distribution. The can quickly mess up the approximated distribution and stir the complete exploration of the guiding process in wrong directions.

Clamping the radiance/power values for each sample would solve the problem of the fireflies having to much influence in the incident radiance distribution fits. On the other hand it would also damp the exploration of important, yet under-sampled regions, which is an important part of the forward-learning approach. Our solution to this problem is to clamp the throughput (e.g. to 10) of the radiance samples while keeping the actual emitted radiance from the light source untouched. In this way samples of high variance which are generated by a bright light source will still have a high contribution and therefore will be further explored and the ones that reached a dark light source will have a lower contribution and will only be further explored in more detail if this low contribution is high relative to the other samples.

Spatial Subdivision

In the following paragraph we explain, how the spatial structure for our guiding caches is build and explain some potential problems, that needs to be considered when building such a structure. In our volumetric guiding approach we use a similar spatial structure as Müller et al. [2017] (see Section 10), which is based on a BSP-Tree. The task of the spatial structure is to provide the guiding algorithm with the necessary guiding estimates (e.g. vMF mixtures of the incident radiance distributions) for each point in the scene. Ideally the spatial guiding structure should fulfill the following criteria:

- Cover the complete region of interest (ROI) of the scene which contributes to the final rendering.
- Return a valid guiding representation at each position in the respective ROI.
- The approximated directional distribution of the incident radiance in a leaf node represents the actual incident radiance function.
- The spatial structure adjusts to the characteristics of the light transport in the scene (e.g. shadows, caustics).

The first point is important since it ensures that the path guiding can be applied along the complete generation process of the path. The ROI defines the bound of the BSP-tree which subdivides the scene into smaller regions/nodes. Each of these leafs contains the directional guiding information for the local region covered by it. For small to medium sized scenes using the bounding box of the complete scene or all volumes contained in the scene is a safe estimate for the ROI. Unfortunately, if the ROI is too big the spatial subdivision structure is to deep, which can lead to a large computational overhead, when querying a tree node. On the other hand, if the ROI is too small, the random path can leave the region, where guiding can be used, and the path generation process would fall back to the common sampling decisions. Since these decisions are most likely to not be non-optimal, they could lead to a large increase in the variance of the estimator. Another problem occurs, when the bounding box of the scene can not be estimated upfront, for example, if the scene contains an infinite medium. In this cases we approximate the ROI by taking the bound of all samples used for training the guiding representation (for forward learning the bound of all samples of the first training iteration). To ensure that these samples really cover the needed region, and to consider the fact, that the guided paths my explore a larger region the bound can be scaled by a small factor (e.g. 1.5 or 2).

After the bound of the guiding region is set, the next important step is to subdivide the space based on the samples collected in the pre-processing step or from the previous training iteration. The important thing to consider, when building or updating the spatial subdivision, is, that in the end each leaf node needs to have enough valid data to make a useful estimate about the directional distribution of the incident radiance. To ensure this we propose small adjustments to the original subdivision procedure presented by Müller et al. [2017]. Similar to their approach we split each node until it contains no more than a given number of samples (e.g. $N = 12K$). To ensure that all these samples actually contain valid data we only consider valid samples, meaning that the power/incident radiance estimate is greater zero. Since photons always contain a power greater than zero, this mainly effects the case of forward-learning.

The second adjustment we propose to not use an uninformed strategy, which splits strictly in a XYZ order and always in the middle. This approach can quickly lead to leaf nodes, which contain a low number of valid samples or more often even none at all. These empty nodes will then not be able to fit/learn any guiding representation and the guiding algorithm may falls back to the common approach with a potential higher variance, which leads to patches of higher variance in the rendered image. A way to avoid this is to track the mean and variance of the positions of all valid samples of a node and split the node a the mean position in the dimension of the highest variance. When using a photon based approach this subdivision method ensures that the number of samples inside a leaf node is guaranteed to almost being close to $N/2$. In a forward learning approach it increases the probability, that in the next training iteration the number of valid samples in the new leaf node is close to N (if the number of samples per pixels is doubled). For volumetric path guiding this extension is important, especially in scene with complex bounding objects and sub surface scattering (e.g. Buddha scene). Here the simple splitting method leads to empty nodes outside the object, which usually are not considered by the volumetric path guiding algorithm. Unfortunately, some of these empty nodes can cross the boundary of the statue a little bit leading to none or an insufficient guiding information in this area. During the rendering process these areas become visible because of high variance noise of the common sampling approach the algorithm felt back to.

While the previously introduced extensions ensure that the first two important characteristics of a guiding structure are fulfilled, it is hard to guarantee the third and the fourth one and we still consider it as an open problem. At the moment in all cached based guiding algorithms the data used fit the directional distribution of the incident radiance is dependent on the maximum number N of samples (valid or invalid) allowed per leaf node. This number is a heuristic guess, based on the assumption that a high enough number of samples will also increase the probability that enough valid information about the incident radiance distribution is contained inside a leaf node. On the other hand a too large value for N prohibits the spatial structure to refine and therefore get more local estimates for the incident radiance. In practice we observed, when using the forward learning approach, that in complex scenarios, where the main contribution of the illumination of a scene comes from hard to sample sources (complex caustics, refracted physical correct sun models), the number of samples in the scene containing information about this source is rather low. If N is too small and the tree subdivides too fast, these samples are missed and the contribution of these sources is neglected by the used path guiding method. Fortunately this problem mainly occurs in extremely complex lighting scenarios, and are often not happening in common scenarios, but we believe it is still important to point out their existence.

Mixture Model Fitting

In the this paragraph we are going to explain, how the set of samples clustered inside a node of the spatial structure can be used to fit the vMF mixture representation of the incident radiance estimate for the corresponding node.

Similar as for the Gaussian mixture models (GMMs) used by Vorba et al. [2014] the vMF mixture models can also be fitted via an expectation maximization (EM) algorithm. The goal of EM is to fit the components of the mixture in a way that the represent the distribution of samples used for the fit. To compensate for the fact, that the directional distribution of the our samples (e.g. photons or incident radiance samples) does not correspond to the actual shape of the spherical incident radiance sample, we also use the weighted EM extension of Vorba et al. [2014]. By adding weights to each samples their overall distribution is biased to simulate a distribution based on the incident radiance function. For further details about the derivation of the weighted EM and how it can be adjusted to the vMF mixtures we refer the reader to the papers of Vorba et al. [2014] and Herholz et al. [2019].

The only difference between fitting the vMF mixture from the different sample types is the way, how the weight for each sample is calculated. For photon based samples, where the directional distribution is almost according the incident radiance distribution the correction weight is the power corresponding to the photon:

$$w_j = \Phi_p(x_j, \omega_j). \tag{33}$$

In the case of the sample being an incident radiance estimate form the forward learning approach the weight has two functions, it first must compensate for the fact, that the directional distribution of samples does not correspond with the actual distribution of the incident radiance, and second it has to bias the samples to simulate this distribution. The later is done by using the incident radiance estimate as weight, while first is achieved by dividing the estimate by the PDF of sampling this direction. The resulting weight for a sample based on the incident radiance estimates from the forward learning approach therefore resolve to:

$$w_j = \frac{\tilde{L}(x_j, \omega_j)}{p_\omega(x_j, \omega_j)}. \tag{34}$$

Using these kind of weights for fitting the vMF mixture for the incident radiance estimates, leads to a mixture, where the weight of each component directly corresponds to the relative amount of fluence/flux represented by that component.

While the representation of the incident radiance distribution via vMF mixtures and their features to build the product or convolution distribution with the phase function are a necessity for making guiding the volumetric transport possible, their fitting based on EM comes with a computational overhead compared to the QuadTree representation of Müller et al. [2017]. This overhead scales linearly with the number of samples per mixture, the number used mixture components or used EM training iterations. The two ways to keep the computational overhead low is either to keep the numbers of samples, components or training iterations low, or to optimize the code of the vMF evaluation and EM fitting. For the latter we made extensive use of implementing all algorithms using current vector engines such as SSE, AVX or AVX512. This way we are able to evaluate 4, 8 or even 16 components at the same time. Since the exponential function is used quite often and is relative expensive to evaluate, we use the same fast approximate version of it as provided by code of evaluating the GMMs by Vorba and Křivánek [2016]. In addition making sure that all the data used for the fitting lies in memory in a cache friendly way also helps increasing the efficiency of the EM code.

In practice we found that for volumetric light transport between 8 and 16 components are sufficient to represent the incident radiance distribution, and that the EM ends up with a stable fit at around $20-30$ iterations.

Estimating Fluence

The fluence/flux of a distribution is an important quantity for our volumetric path guiding approach. Its main purpose is to scale the vMF mixtures representing the distributions of the incident and in-scattered radiance, so that their queried values represent the estimates of the volumetric light transport quantities.

It is assumed that the fluence/flux in constant in each node of the spatial structure. Its value can be estimated from the samples used to fit the incident radiance distribution for each node. Based on the use learning methods and the corresponding type of samples used to fit the distributions the formula for estimating the fluence changes.

For radiance sample the estimation of the fluence is rather simple and can be calculated by using the incident radiance estimates and the sampling PDF for the direction of the samples to build a Monte-Carlo estimator for the fluence:

$$\Phi(x) = \frac{1}{N}\sum \frac{\tilde{L}(x_n, \omega_n)}{p_\omega(\omega_n)}. \tag{35}$$

It is worth noting that this formula assumes that the samples also contain the ones, where $\tilde{L} = 0$. If only the valid samples are stored it is also necessary to track the ratio for each leaf node of the valid and overall drawn samples and correct the fluence estimate by multiplying it with this. In the case that samples are generated via a photon tracing a volumetric density estimation step has do be performed to convert the power of the photons into fluence. For this, it is necessary to know the volume covered by the photons. Unfortunately, the volume of the nodes of the spatial structure are not sufficient enough for this task. The space that is covered by a node of the structure my only cover a part of the actual volume and therefore would overestimate the actual volume of the photons. A simple way to estimate the volume, which might not be the optimal one, is to use the one of the smallest bounding box around the samples. More advanced method would be to approximate the convex hull of the samples but where not further evaluated by us. The formula for estimating the fluence from a set of photon samples is given by:

$$\Phi(x) = \sum \frac{\Phi_p(x_n, \omega_n)}{V(x_1, \ldots, x_N)} \tag{36}$$

It is worth noting, that the fluence/flux calculation using photons for surfaces is based on the surface area covered by the photons, which might the a reason, why it is not straight forward to merge volume and surface photons to build uniform incident radiance guiding representation for surface and volume guiding.

11.5.2 Phase Function Representation

A crucial part of our volumetric path guiding approach is the representation of the phase function via a vMF mixture. The quality of the representation influences each guiding decision, since it is needed for sampling directions according the product distribution of the phase function and the incident radiance and to estimate the in-scattered radiance for the distance sampling and directional splitting procedures. In this section we are going to explain our method to robustly fit various types of phase function in a pre-processing step, while later we will explain how the representation can be extended to support varying parameter for a given phase function model. The sections ends with a discussion about the problematic, which can arise in complex production environments, where the shape of the phase can be unknown upfront.

Fitting

In practice a phase function can be a combination of multiple lobes, of multiple phase function type/models (e.g. Henyey-Greenstein, Rayleigh, Cornette-Shanks). To fit a vMF mixture for a given phase function we applied a small pre-processing step at the beginning of the rendering process. For each volume in the scene we fitted a vMF mixture suing a non-linear optimization framework such as Ceres Agarwal et al. [2016]. Using a set of N uniform samples over azimuth angle θ we fit the vMF mixture parameter Θ_f for that phase function:

$$\underset{\Theta_f}{\arg\min} \sum_{n=1}^{N} \left[L_{\log}(f(\omega_n, \ldots), \mathrm{V}(\omega_n, \Theta_f)) \right]^2 \tag{37}$$

When fitting a phase function, one has to be careful in selecting the right loss function for the optimization, so that all the features of the phase function are represented well enough by the mixture used for sampling it. Considering the standard Henyey-Greenstein (HG) phase function, which shape on the first sight looks quite similar as a vMF lobe, it has a long tail, which leads to an important back scattering characteristic even for high mean cosines. The used loss function therefore, needs to take care to reliably represent this long tails, while still focusing on the peak of the phase function. We found that following loss function lead vMF mixture parameters,that can robustly represent a variety of phase function:

$$L_{\log}(d, m) = \log(d + \varepsilon) - \log(m + \varepsilon), \tag{38}$$

where

$$\varepsilon = 0.0001 \cdot \max(d_1, ..., d_N). \tag{39}$$

This loss function is similar to the logarithmic version of the relative loss function, that resolves to zero, if $d/m = 1$. The additional epsilon is used to damp the focus of the optimization on the tail of the phase function, in cases that the phase function is highly forward/backward scattering (e.g. HG $g > 0.95$). In our experiments using a mixture of 3 to 4 vMF lobes is enough to represent a variety of different phase functions. In the case that the mean cosine values of a given model vary through the volume, and the number of mean parameters of a mixture of phase function lobes is low (e.g. 1 to 2) it is possible to leverage the manifold representation of the mixture parameters, as introduced by Herholz et al. [2018] or to tabulate the parameter space in the pre-processing step.

Problematic Production Cases

The pre-processing step relies on the fact that the shapes of used phase functions in a scene is known upfront, and that their number is limited, to be able to fit the vMF mixture representations in usable time or memory budget. In cases, where mixtures of M lobes and their parameters (e.g. mean cosines, mixture weights) are defined as heterogeneous volume parameter or even as procedural function it is hard to predict all possible combinations upfront. One solution would be to fit a vMF mixture for each possible lobe type and its mean cosine parameter, and then build a mixture vMF mixtures on the fly. Unfortunately, this would quickly increase the number of components needed to represent one instance of a phase function (e.g. $4xM$), which has a direct impact on the efficiency of our guided sampling decisions. A proper solution for this problem is an open question, which needs further investigation.

References

Sameer Agarwal, Keir Mierle, and Others. 2016. Ceres solver. http://ceres-solver.org.

Thomas Bashford-Rogers, Kurt Debattista, and Alan Chalmers. 2012. A significance cache for accelerating global illumination. *Comput. Graph. Forum* 31, 6 (2012), 1837–1851.

Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. 2013. Joint importance sampling of low-order volumetric scattering. *ACM Trans. Graph.* 32, 6 (2013), 164:1–164:14.

Sebastian Herholz, Oskar Elek, Jens Schindel, Jaroslav Křivánek, and Hendrik P. A. Lensch. 2018. A Unified Manifold Framework for Efficient BRDF Sampling based on Parametric Mixture Models. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*, Wenzel Jakob and Toshiya Hachisuka (Eds.). The Eurographics Association. https://doi.org/10.2312/sre.20181171

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product importance sampling for light transport path guiding. *Computer Graphics Forum* 35, 4 (2016), 67–77.

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik Lensch, and Jaroslav Křivánek. 2019. Volume Path Guiding based on Zero-Variance Random Walk Theory. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019)* X, X (X 2019).

J. Eduard Hoogenboom. 2008. Zero-variance Monte Carlo schemes revisited. *Nuclear Science and Engineering* 160, 1 (2008), 1–22.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob. 2012. *Numerically stable sampling of the von Mises-Fisher distribution on S^2 (and other tricks)*. Technical Report. Cornell University.

James T. Kajiya. 1986. The rendering equation. *SIGGRAPH Comput. Graph.* 20 (1986).

Christopher Kulla and Marcos Fajardo. 2012. Importance sampling techniques for path tracing in participating media. *Comput. Graph. Forum* 31, 4 (2012), 1519–1528.

Jaroslav Křivánek and Eugene d'Eon. 2014. A zero-variance-based sampling scheme for Monte Carlo subsurface scattering. In *ACM SIGGRAPH 2014 Talks*. 66:1–66:1.

Johannes Meng, Johannes Hanika, and Carsten Dachsbacher. 2016. Improving the Dwivedi sampling scheme. *Computer Graphics Forum* 35, 4 (2016), 037–044.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum* 36, 4 (2017), 91–100.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37, 2 (2018).

Jan Novák, Iliyan Georgiev, Johannes Hanika, Jaroslav Křivánek, and Wojciech Jarosz. 2018. Monte Carlo Methods for Physically Based Volume Rendering. In *SIGGRAPH 2018 Courses*. Article 14, 1 pages. https://doi.org/10.1145/3214834.3214880

Vincent Pegoraro, Ingo Wald, and Steven G. Parker. 2008. Sequential Monte Carlo integration in low-anisotropy participating media. In *Proceedings of the Eurographics Conference on Rendering*. 1097–1104.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Ph.D. Dissertation. Stanford University.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph.* 33, 4 (2014), 101:1–101:11.

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-driven Russian roulette and splitting in light transport simulation. *ACM Trans. Graph.* 35, 4 (2016).

12 Guiding in Path Space

Johannes Hanika, *Weta Digital*

In this section we will take the ideas presented in the last section, in particular volume sampling and Russian roulette, and relate them to guiding of full paths in path space [Simon et al., 2018]. Guiding new samples along full guide paths instead of marginalised distributions which only guide low dimensional parts at a time transparently includes all aspects of the high dimensional path space such as path length, BSDF, incident illumination, and free distances in volumes. It also allows us to use simpler, uni-modal functions to represent a continuous PDF around guide samples. However both marginalised caches and full paths come with advantages and drawbacks.

12.1 Introduction

We have seen in the previous sections how it is important to sample all aspects of a transport path to effectively reduce variance. This naturally includes incident illumination, BRDF or phase function, and distances in volumes. Combining these from caches which store 2D marginals (for instance only incident illumination or only BSDF) may incur multiplication of these stored signals. This may turn out to be expensive, but in full fledged production renders sometimes compute on that scale can still be hidden between the dominating memory access costs caused by shading or ray tracing. In any case it will incur another reduction of accuracy: both the incident illumination and the BSDF fit will come with errors, which accumulate when multiplying the signals.

There are also more subtle aspects to consider when sampling a new path: the input samples know what kind of event happened. In the simplest case, this may be a reflect vs. a transmit event at a dielectric surface, for instance the cornea of a character's eye. Since the Fresnel terms tell us that most light will be transmitted, it is almost always a good idea to transmit a lot of light, to render a noise free image of the iris behind the cornea. However, there are a few rays that captured the direct highlight on the cornea. These may remain noisy for a long time in practice when importance sampling the low value of the reflecting Fresnel term. Previous samples carry the information whether reflect or transmit was a good idea at a specific point in path space. When projecting down such information to 2D directional distributions, it is often lost.

Similar holds for Russian roulette: how long should a path be in certain regions of the scene? This information is contained in the set of previously traced paths. Resampled by their throughputs they represent a mixture of path lengths with spatial resolution which tells us exactly how long we expect a path to be.

This leads us to another path guiding method, based on caching all this information in its entirety: *path space guiding* [Simon et al., 2018]. In this approach, no information about previous guide samples is discarded, full paths are retained to construct a sampling density from. To make the memory usage tractable, we need to select guide samples only where necessary, i.e. where the underlying base sampler (such as simple path tracing with next event estimation) does not yet yield a low variance estimator.

12.2 Overview

In fig. 30, an example integrand $f(\mathbf{X})$ is shown in blue, and the existing MC estimator draws samples from the PDF shown in orange, which we denote as the *unguided PDF* $p_u(\mathbf{X})$. More formally, we are approximating the integral using the estimator:

$$\langle I(\mathbf{X})\rangle_u = \frac{f(\mathbf{X})}{p_u(\mathbf{X})}. \tag{40}$$

In this example, the PDF p_u represents the left smooth mode of the integrand well, but misses the other features (two more modes). Thus we construct the additional *guided PDF* $p_g^i(\mathbf{X})$ (shown in green) which is iteratively refined for $i = 0, 1, \ldots$, and captures the differences between the integrand $f(\mathbf{X})$ (blue) and

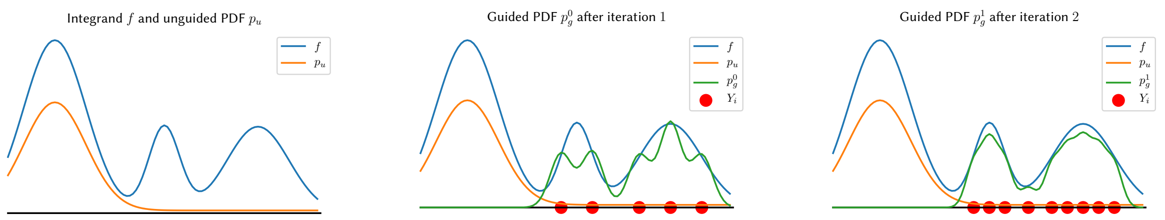


Figure 30: 1D illustration of path space guiding (reproduced from [Simon et al., 2018, Fig. 2]): parts of the integrand (f, blue) are well represented by a PDF (p_u, orange). We wish to approximate the difference between the integrand and the orange PDF. Guiding records ($\mathbf{Y}_i$, red points) are iteratively placed and then used to reconstruct a continuous guided PDF (p_g, green). These two PDF are then combined with multiple importance sampling (balance heuristic) to yield a good Monte Carlo estimator for the integral.

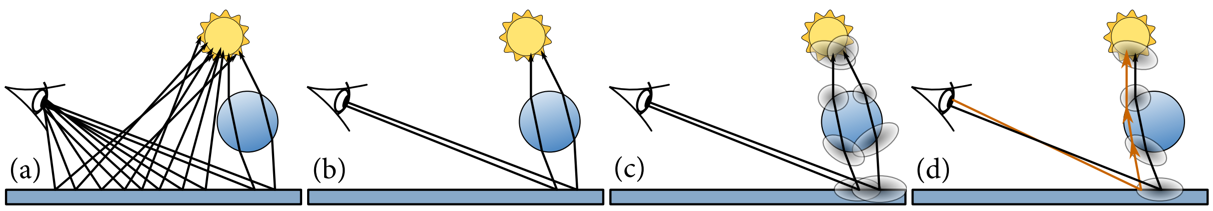


Figure 31: Schematic overview of the guiding method employing full paths, reproduced from [Simon et al., 2018, Fig. 3]. We start with all path traced trajectories, depicted in (a). From these, we select the ones that are most poorly sampled by the underlying Monte Carlo estimator, shown in (b). For each of these guide paths, we compute anisotropic reconstruction kernels at every path vertex, to fill in the gaps between the samples. This is visualised in (c). To sample from this structure, we first pick a guide path and then successively sample path vertices according to the reconstruction kernels (d). Evaluating the PDF requires to sum up the PDF associated with all guide paths that may create the given path.

the unguided PDF $p_u(\mathbf{X})$ (orange). The converged guided PDF is shown in fig. 30, right. Eventually, we want to combine these two PDFs using multiple importance sampling (MIS). More precisely, we will be using a single-sample model with the balance heuristic [Veach, 1998, chapter 9] resulting in a combined estimator with a mixing weight u, i.e.

$$\langle I(\mathbf{X})\rangle_g^i = \frac{f(\mathbf{X})}{u \cdot p_u(\mathbf{X}) + (1-u) \cdot p_g^i(\mathbf{X})}. \tag{41}$$

The guided PDF will be constructed incrementally and with every iteration i we adjust $p_g^i(\mathbf{X})$ to further reduce the variance of eq. (41).

Intuitively, we proceed as illustrated in fig. 31. We begin by letting $p_g^0(\mathbf{X}) \equiv 0$ and effectively sampling $\mathbf{X}$ using only $p_u(\mathbf{X})$ (fig. 31a). From the paths $\mathbf{X}$, we select only a few new guide paths $\mathbf{Y}$ motivated by importance sampling: we first determine whether a path is an outlier causing high variance (via *density-based outlier rejection* [DeCoro et al., 2010]) and only from these we pick the N with the highest contribution to the estimate in eq. (41). We will iteratively add batches of guide paths $\mathbf{Y}_j$ sampled from both p_u and the current guided pdf p_g^i to the cache, and once they are added we keep them unchanged until the end. This is illustrated in fig. 31b and detailed in section 12.3.

The guide paths $\mathbf{Y}_j$ are turned into a continuous function using a Gaussian reconstruction kernel (fig. 31c). We use a high-dimensional neighbour search to determine large enough reconstruction kernels to close the gaps between paths and to achieve smooth coverage. One challenge is that the PDF evaluation can become slow in Gaussian mixture models. Therefore we use truncated Gaussian kernels to be able to cull away samples efficiently.

To sample from the cache, we first select a guide path $\mathbf{Y}_j$ using a *cumulative density function* (CDF) built on path weights w_j which are updated every iteration i to reflect the new guided PDF $p_g^i(\mathbf{X})$.Then

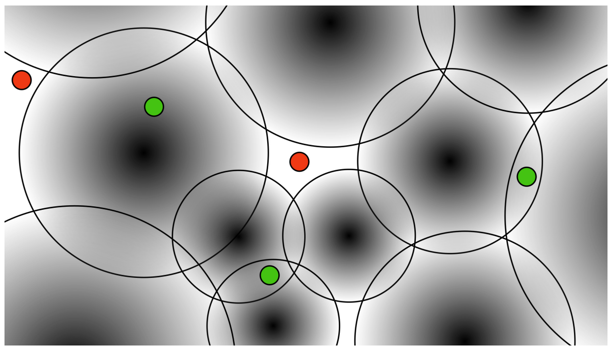


Figure 32: Placing new samples far apart from other samples is essentially a dart throwing method, which can be used to generate blue noise samples. While the guide path sampling does not explicitly search for minima, it increases the probability in these areas: a sample marked in red will have low PDF, resulting in a firefly sample if the function evaluation is high. Such a sample is likely to spawn a new guide path in the next iteration.

we sample a new path vertex $\mathbf{x}_v$ following the Gaussian reconstruction kernel $(\Sigma, \mu)_v$ around every guide path vertex $\mathbf{y}_v$ (fig. 31d), starting at the sensor (see section 12.4).

In the subsequent iteration $i + 1$, we draw samples from both $p_u(\mathbf{X})$ and $p_g^i(\mathbf{X})$, with probabilities u and $1 - u$, respectively. No matter which technique was used, new samples will be considered to be recorded as new guide samples for the next iteration, depending on their contribution $\langle I(\mathbf{X})\rangle_g^i$.

Because $p_g(\mathbf{X})$ does usually not cover the full domain, we require $p_u(\mathbf{X})$ to form a complete unbiased estimator, in particular $\forall \mathbf{X} : f(\mathbf{X}) > 0 \Rightarrow p_u(\mathbf{X}) > 0$. This will not be the case for the guided PDFs $p_g^i(\mathbf{X})$ which may be zero in areas where the original estimator is deemed sufficiently good already. We chose the balance heuristic to combine the PDFs as it is close to optimal [Veach, 1998, chapter 9]. The precise choice of combination heuristic does not affect the final sampling quality much, as the convex combination of both PDFs $p_u(\mathbf{X})$ and $p_g^i(\mathbf{X})$ adapts to be proportional to $f(\mathbf{X})$.

12.3 Selecting guide paths

Guide paths are expensive: first, we need to store them, so we want to keep their overall number to a minimum. Second, PDF evaluation scales somewhat with the number of overlapping Gaussians. Fortunately the culling due to the truncated Gaussians usually works really well. Some of this is due to how the Gaussians are constructed: to span only a certain number of neighbours.

Still we want to carefully select guide paths. This is done by looking at a screen space projection of the noise: density-based outlier rejection (DBOR) [DeCoro et al., 2010] can tell us how lonely is a sample in screen space. This is different to being lonely in high dimensional path space (in a way every sample will be lost in space there). As it happens this is exactly what matters for the final image.

In 2D image space this has an interesting connection to dithering methods which will create blue noise patterns [Ulichney, 1993]: Figure 32 shows a 2D illustration. Sampling driven by Gaussians around existing samples will gather probability mass around existing samples. In areas right in the middle between samples, there will be a low probability density. If the function value is high in these areas, the probability to sample an outlier here is increased. This means that, in a stochastic sense, such a sample placement is similar to dart throwing.

12.4 Sampling new paths

Sampling paths is the easy part in this setup. Conceptually we can simply choose a guide path, and then successively sample the next transport vertices starting from the eye. It is possible to support volumes natively in this context, but there are a few details as to including the BSDF for (near) specular events (please see the original paper by Simon et al. [2018]).

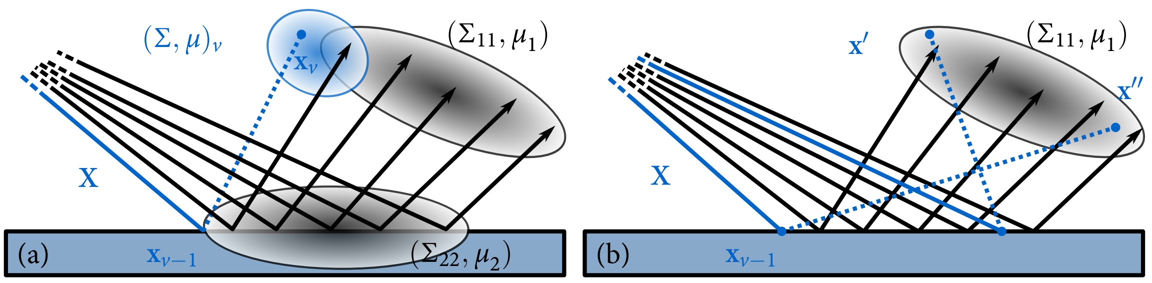


Figure 33: Illustration reproduced from [Simon et al., 2018, Fig. 7]. For most accurate sampling of $\mathbf{x}_v$, we compute a covariance matrix (5×5 in this example, since $\mathbf{x}_{v-1}$ is a surface point and $\mathbf{x}_v$ is in a volume). One block in the matrix, Σ_{11}, expresses how $\mathbf{x}_v$ should be distributed according to the collected data from the guide paths (shown in black). Because we already have a fixed previous vertex $\mathbf{x}_{v-1}$, we derive the conditional of the 5×5 Gaussian accordingly, resulting in a new distribution $N[\Sigma_v, \mu_v]$ (blue). If there is a non-zero covariance between the coordinates of the guide path vertices $\mathbf{y}_v$ and $\mathbf{y}_{v-1}$ the conditional Gaussian Σ_v will typically be a lot more focused than Σ_{11}. In this example, disregarding the covariance would amount to sampling according to Σ_{11} with much larger spread, as illustrated in (b). Using Σ_v results in higher quality samples which are more likely to be valid paths, and faster PDF evaluation due to culling.

One key aspect is that the Gaussian stored on a transport vertex is 4D (in the surface-surface case). That is, it includes information about covariance between the current vertex on the surface and the next vertex that we are about to sample. This facilitates some parallax correction: if we arrive at the surface some distance away from the vertex of the guide path, the 4D distribution will be conditioned on the current vertex. This shifts the mean of the remaining dimensions and overall results in a more precise distribution (see fig. 33). In particular, it avoids the grid artifacts which can easily arise in the other guiding methods which store a directional distribution per voxel (see section 10.4 where the same artifact is avoided by different means).

12.5 Determining the size of the Gaussians

The guide paths are equipped with Gaussian distributions at every vertex. These are usually 4D (from a point on the surface to another point on the surface). To construct them, we need to fit the distribution to a few samples. We can use the neighbouring guide paths for this, using a high dimensional nearest neighbour search: every vertex adds three dimensions. This can work well, but has two pain points: one, a high dimensional neighbour search can be slow and is hard to accelerate. We can usually cull based on a few things (bounding boxes of truncated Gaussians, path lengths and configurations etc), but building acceleration structures hardly pays off for 1000s of paths with 100s of dimensions. Second, the results depend on the number of neighbours we choose to use. The estimation is stable with a minimum number of neighbours (such as 10 or 20) but will change if we go to the 100s. If we go too far, we run the risk of evaluating neighbours which do not belong to our unimodal lobe around the guide path any more.

12.6 Discussion

While, as discussed in the previous section, the estimation of the distribution based on neighbours has some issues, it is clearly preferable over using analytic derivatives: neighbours are an estimate of the real data in the surroundings, including BSDF, displacement, curvature, complex shadowing, etc.

This guiding technique comes with a built-in dependency on density-based outlier rejection. Similar to variance-based reweighting (see section 10.3) this makes sure we can always present intermediate images to the user, and not throw away information for successive progressions. The DBOR-as-post-process approach also makes it possible to show noise free results at all stages. This comes at the cost of bias, but since the data is still around, an unbiased estimate can be constructed if needed.

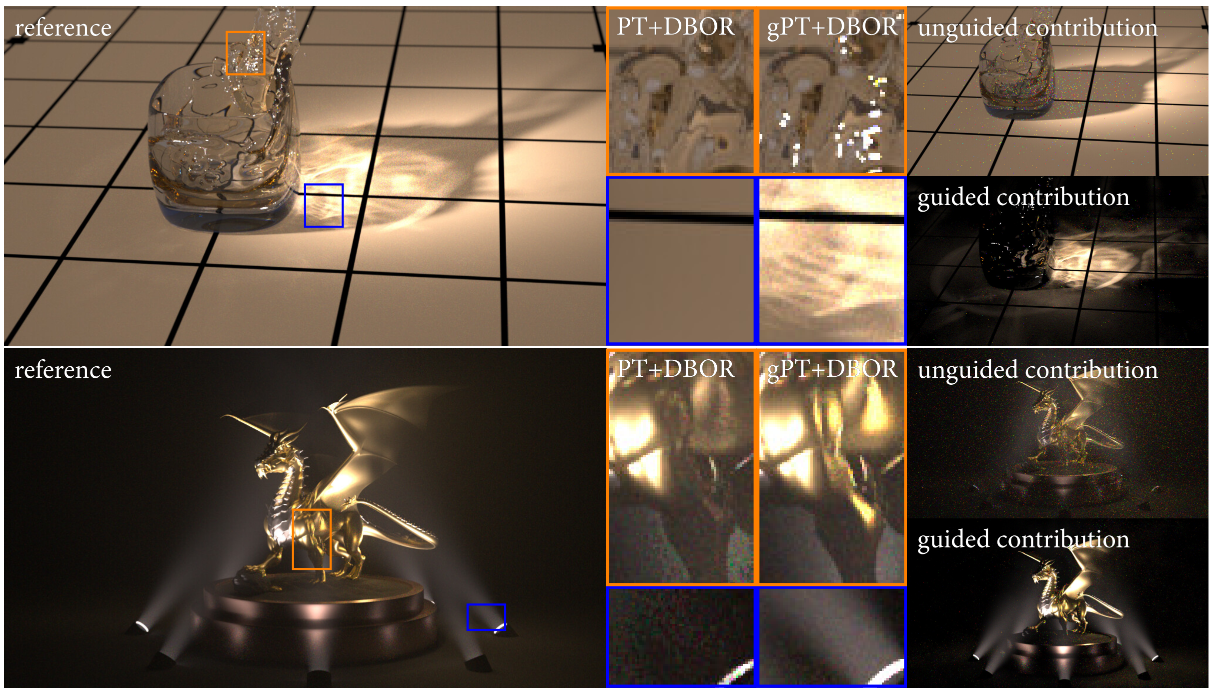


Figure 34: Equal-time comparisons of path tracing (PT) and path space guided path tracing (gPT). The insets show PT and guided PT with outlier removal and therefore PT misses lighting effects it can not render efficiently (e.g. caustics). In all scenes guided PT robustly identifies the part of light transport that can not be handled by PT efficiently and improves upon this using guided sampling only for these parts. As a side effect, we obtain an interesting separation into a smooth/low contrast image (unguided contribution) and a sharp/high contrast image (guided contribution), which could be denoised separately.

The second aspect is that DBOR only selects paths which are hard to sample without guiding. This is similar to section 10.5 and means the method can stay cheap where it's not needed.

Sampling from a fixed guide path is very fast. Evaluating the Gaussians touches very little data and is straight forward to do. On the flip side, evaluating the PDF requires to touch all guide paths with overlapping Gaussians. This needs careful implementation and efficient culling to result in fast execution.

Keeping the full paths around is the most basic way of storing information about previous samples. There is clearly nothing important lost in the process, because all data is still there. This makes it simple to visualise what is going on for debugging, or reproject guide paths for a different frame in an animation. This is also facilitated by another nice property of the path guiding cache: we can hand over just a path for inclusion in the cache, without a PDF. It will be evaluated based on how likely it is to construct such a path given the sampling density that is already contained in the cache. This way, we can even evaluate invalid samples which come from resampling the last frame, or from an adjoint transport operator (or even hand drawn samples if there is an application for that).

Another advantage of storing full paths is that this representation transparently includes all aspects of sampling: path length (Russian roulette), BSDF, distances in volumes, reflect vs. transmit choice etc.

Unfortunately exactly this full-path property also has a downside. This is especially apparent for multiple scattering: here the exact high dimensional path configuration plays not much of a role. It is only important to exit the volume towards the light source. This is an instance of the *curse of dimensionality*: we would require a number of guide paths growing exponentially with the number of dimensions. This quickly becomes intractable if the sampling space cannot be broken down into small and directed problematic regions. These become essentially low-dimensional and can be approximated by full paths very well. Unfortunately this is a big assumption to make, so the technique cannot be switched on by default.

References

Christopher DeCoro, Tim Weyrich, and Szymon Rusinkiewicz. 2010. Density-based Outlier Rejection in Monte Carlo Rendering. *Computer Graphics Forum (Proc. Pacific Graphics)* 29, 7 (Sept. 2010), 2119–2125.

Florian Simon, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2018. Selective guided sampling with complete light transport paths. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Dec. 2018). https://doi.org/10.1145/3272127.3275030

Robert A. Ulichney. 1993. Void-and-cluster method for dither array generation, Vol. 1913. https://doi.org/10.1117/12.152707

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation.* Ph.D. Dissertation. Stanford University. AAI9837162.

13 Open Problems and Future Work

Jirí Vorba, *Weta Digital*
Johannes Hanika, *Weta Digital*

We believe that there is still an unexplored potential of guiding methods to further reduce number of samples required for rendering clean images. One of the main goals of this course is to identify the most pressing problems and share them with the research community and thus to enable further exploration of path guiding. We also discuss the possibility of combining some of these methods so that the best of them would form more efficient algorithms.

Types of cache records The methods we explored in this course use a variety of different encoding schemes to represent the distributions of samples. Even the dimensionality of such directional caches is subject for further research: is a 2D incoming light field enough, if multiplied by BSDF? Can it be unified to include distance sampling, too? 4D distributions have been shown to give some benefits when using the conditional distribution to interpolate for different shading points. Using all dimensions in a path completely brings some benefits (for instance for Russian roulette) but quickly exhausts the limits of tractable computation due to the curse of dimensionality.

Accuracy of cached distributions Naturally we are striving for the most accurate importance sampling scheme and thus want a perfectly precise representation of the 4D plenoptic function. However, this is a much harder problem to solve than just computing an image: the image is 2D and very limited in extent. The scene may be huge and computing a 4D function everywhere in it is computationally more demanding than ”just” computing an image. Thus, there has to be a trade off here, learning just what is necessary to converge the image faster. This opens the questions what the appropriate basis for a distribution would be: Gaussian mixtures, radial basis functions, or neural networks?

Can we switch it on by default? One argument for adoption in production environments is whether the technique can always stay switched on without the user worrying about the implications. This means the technique needs to be invisible, i.e. not make easy scenarios slower by bloaty computations in the background. It also means there should not be any edge cases where it performs much worse than other techniques.

Raw speed In general we can invest a lot of time in computing good samples in offline rendering. Many of our integration problems are so hard that we cannot hope to converge them with standard sampling methods. That is, we require many thousands of samples, at which point the dreaded $1/\sqrt{N}$ convergence rate of Monte Carlo has stalled completely in practice. Here, only higher quality sampling with lower standard deviation can help. There are, however, two things to keep in mind. First, we are searching for an *always-on* method that will transparently do the best it can without slowing down simple cases. Second, the square root tells us that we cannot be more than quadratically slower in generating samples at half the standard deviation. Some guiding methods can be so expensive that a simpler but worse method will outperform them by drawing many more samples in the same time.

One future direction here might be to explicitly pick sampling techniques based on their performance based on the scene at hand.

Generality To result in a maintainable solution that will find adoption in practice, we want to find an algorithm that is simple yet general: that is, we don’t want to code special cases for volumes, for surfaces, for dense scattering, for godrays in thin fog, if possible. This is to date the weakest point of the toolset we have at our disposal for rendering. Only once this issue is resolved can we start to think about low level optimisations.

Robustness All adaptive sampling techniques run the risk of being worse than uniform sampling in some cases: if important areas remain undiscovered, but a spike is detected somewhere, the algorithm might explore only this spike, neglecting the task of discovering the other areas. Most guiding methods take explicit countermeasures against this case by always mixing in some portion of independent sampling, regularising the distributions in the beginning, or by employing the first and the second moment (for instance by summing the variance to the importance of some slot [Vévoda et al., 2018]).

Neural Path Guiding Neural networks applied to surface path guiding were demonstrated to yield state-of-the-art quality [Müller et al., 2018], albeit at prohibitive computational cost that requires high-end GPU(s) for practical performance. To bridge the performance gap and to increase quality further, it would be interesting to investigate alternative approaches [Chen et al., 2018, Huang et al., 2018], cheaper neural networks [Keller et al., 2019], and more advanced optimization algorithms [Izmailov et al., 2018]. Additionally, it would be interesting to apply the neural approach to previously difficult higher-dimensional settings such as temporal path guiding for motion blur and distance sampling in volumes with non-exponential transmittance [Bitterli et al., 2018].

References

Benedikt Bitterli, Srinath Ravichandran, Thomas Müller, Magnus Wrenninge, Jan Novák, Steve Marschner, and Wojciech Jarosz. 2018. A radiative transfer framework for non-exponential media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (nov 2018), 225:1–225:17. https://doi.org/10.1145/3272127.3275103

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural Ordinary Differential Equations. *arXiv:1806.07366* (June 2018).

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron C. Courville. 2018. Neural Autoregressive Flows. *arXiv:1804.00779* (April 2018).

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv:1803.05407* (March 2018).

Alexander Keller, Matthijs Van Keirsbilck, and Xiaodong Yang. 2019. Structural Sparsity: Speeding Up Training and Inference of Neural Networks by Linear Algorithms. In *GTC Talks*. https://on-demand-gtc.gputechconf.com/gtcnew/sessionview.php?sessionName=s9389-structural+sparsity%3a+speeding+up+training+and+inference+of+neural+networks+by+linear+algorithms

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. 37, 4 (Aug. 2018), 125:1–125:12.