Bachelor's Thesis

Bidirectional Neural Radiance Caching

Julian C. Stamm



Institute of Computer Science II – Visual Computing
University of Bonn

First Reviewer: Prof. Dr. Reinhard Klein
Second Reviewer: Prof. Dr. Florian Bernard

September 11, 2025

Summary

This thesis explores the topic of online radiance caching in the context of real-time rendering. Therefore, I will leverage the potential of neural networks to learn representations of the radiance field as proposed by Müller, Rousselle, et al. (2021).

Furthermore, I propose a theoretical abstraction of radiance caching, which I use to introduce three novel training techniques that leverage light tracing to improve the radiance cache's performance for difficult light paths including caustics and indirect illumination.

To achieve real-time performance, I provide an efficient implementation leveraging modern graphics hardware features such as hardware accelerated ray tracing and tensor cores. I compare my results against reimplementations of the original method by Müller, Rousselle, et al. (2021) and the follow-up work by Müller, Evans, et al. (2022), comparing both performance and visual quality.

Acknowledgments

I want to thank my advisor, Tom Kneiphof, for his precious guidance and his invaluable insights and feedback. I also want to thank my family and friends for their constant backing and accompaniment throughout my studies. Special thanks, I want to give to Alexa for her unwavering encouragement and unconditional support.

Disclosure of Generative AI Usage

In appreciation of the increasing applicability of Generative AI in academic work, I openly acknowledge its utilization in preparation of this thesis. Generative AI tools were used to help reword and polish paragraphs to improve readability and ease of understanding. Generative AI tools also helped in researching applicable libraries and technology, as well as interpreting various concepts covered here. All work created or edited through Generative AI has been independently checked and approved by me as true and valid. I claim sole responsibility for the content and attest that the end-product represents my own comprehension and effort. No content or output from any Generative AI was used verbatim without critical evaluation and rewording. All sources, human as well as AI-generated, have been cited where applicable and verified and checked on truthfulness. Generative AI was used as a supportive tool only and never as a substitute for original research, analysis, or critical thinking involved in this thesis.

Contents

1 Introduction **1**

2 Related Work **3**
2.1 Global Illumination (GI) . . . 3
2.2 Path Reuse . . . 4
2.3 Neural Rendering . . . 6
2.4 Prerequisites . . . 7

3 Constructing a Reference Pathtracer **9**
3.1 Solving the Rendering Equation . . . 9
3.2 Monte Carlo Integration . . . 9
3.3 Randomized Quasi-Monte Carlo Integration . . . 10
3.4 Principled BSDF . . . 11
3.5 Importance Sampling . . . 13
3.6 Evaluation of the Weighted BSDF . . . 15
3.7 Multiple Importance Sampling . . . 15

4 Neural Radiance Caching **17**
4.1 Network Architecture . . . 18
4.2 Input Encodings . . . 18
4.3 Training . . . 22
4.4 Inference . . . 23

5 Bidirectional Radiance Caching **27**
5.1 A General Theoretical Framework for Radiance Caching . . . 27
5.2 The Path Space Integral Formulation . . . 28
5.3 Inference . . . 30
5.4 Path Tracing . . . 31
5.5 Bidirectional Training (BT) . . . 32
5.6 Light Training (LT) . . . 34
5.7 Sparse Progressive Photon Collection (SPPC) . . . 35

6 Results **41**
6.1 Methodology . . . 41
6.2 Parametrizing the NRC . . . 42
6.3 Optimizations . . . 44

6.4 Evaluation 46

7 Conclusion **51**
7.1 Summary 51
7.2 Further Optimization 51
7.3 Future Work 52

Bibliography **55**

1 Introduction

A longstanding and essential problem in computer graphics is Global illumination (GI), which is the simulation of both direct and indirect lighting. Many algorithms were developed to approach this problem, the most prominent and widely used of which is path tracing (Kajiya, 1986) for its simplicity and generality. In recent years, huge advancements in GPU architecture made interactive path tracing feasible. Yet, path tracing is still slow compared to traditional rasterization and suffers from high temporally instable noise. Radiance caching (G. J. Ward et al., 1988) tackles both of these shortcomings. By only estimating radiance sparsely and interpolating in between, reusing radiance estimates from previous frames, radiance caching can significantly reduce the number of samples per pixel while also averaging temporally and spatially near estimates to reduce noise.

Müller, Rousselle, et al. (2021) recently proposed a particularly elegant approach to radiance caching that utilizes neural networks to do the interpolation, leveraging the generalization capabilities of neural networks and the online adaptation capabilities of modern optimizers. Their approach, called Neural Radiance Caching (NRC) estimates the outgoing radiance field with a multilayer perceptron (MLP) that is trained online during rendering. Particularly important for a high quality result is the choice of input encoding, as the MLP is kept shallow for performance reasons, so the input should correlate roughly linearly with the output. Müller, Rousselle, et al. (2021) originally used a simplified version of the Fourier series to linearize space (Tancik et al., 2020). Müller, Evans, et al. (2022) later proposed the Multiresolution Hash Encoding (MHE), which uses feature vectors distributed in hash-grids and significantly improves quality of high frequency details.

However, because the training of NRC is based on unidirectional path tracing, it struggles to capture low-probability high-contribution light paths like caustics and indirect illumination. In this thesis I aim to generalize NRC to enable the use of radiance estimators better suited to these phenomena. I integrate three radiance estimators into the training, one based on bidirectional path tracing (Lafortune and Willems, 1993), one based on light tracing (Arvo et al., 1986) and one based on progressive photon mapping (Hachisuka, Ogaki, et al., 2008).

Contributions To summarize, my contributions are the following:

- I propose a generalized theoretical framework for Neural Radiance Caching by Müller, Rousselle, et al. (2021), allowing for arbitrary and independent modifications to the inference, training and interpolation steps.
- For the training step, I propose three novel training methods:
  - A bidirectional training approach that excels at capturing complex indirect lighting effects.
  - A light tracing technique that has potential to learn sharp features but suffers from instability and strong bias.
  - An adaptation of Progressive Photon Mapping (Hachisuka, Ogaki, et al., 2008; Jensen, 1996) which extends the original technique with sparse online caching and path space denoising and robustly learns both indirect lighting and caustics.

Structure Chapter 2 gives an overview over solutions to the Global Illumination problem with a particular focus on approaches that cache radiance or leverage neural networks. After that, chapter 3 provides a theoretical background on path tracing and derives a physically plausible BSDF. Chapter 4 then introduces Neural Radiance Caching and gives details about the network architecture, input encodings and the training and inference procedure. Chapter 5 contains the main contribution by providing a generalization of radiance caching and derivations of the three proposed radiance estimators. In chapter 6 detailed comparisons of all described techniques are provided. Finally, chapter 7 summarizes the findings and motivates interesting areas for future research.

2 Related Work

The research related to this thesis can be divided into three main categories: First, I will give a short overview over several relevant approaches to the global illumination problem. After that, I will specifically discuss methods that reuse computed radiance estimates across time or space. Finally, I will take a closer look into papers that apply neural networks to rendering. For a more in-depth overview over the whole field, I refer to Ritschel et al. (2012) and for photon mapping in particular to Kang et al. (2016).

2.1 Global Illumination (GI)

In the early days of rendering, lighting was limited to direct illumination and ray traced reflections (Whitted, 1980) and indirect illumination was often hand-crafted (Christensen and Jarosz, 2016).

Finite Element Methods Goral et al. (1984) were the first to also compute indirect illumination, which they achieved by dividing the scene into finite elements and solving the resulting system of equations. Their approach was limited to diffuse surfaces, though later papers extended this idea to non-diffuse surfaces (Immel et al., 1986). However, the separation of the scene into finite elements generally results in visible bias, particularly for complex geometry.

Monte Carlo Methods The first universal unbiased algorithm to solve the global illumination problem was given by Kajiya (1986), who formulated a radiance estimator based on Monte Carlo integration. Simultaneously, Arvo et al. (1986) discovered the potential of light tracing to simulate indirect lighting effects. Lafortune and Willems (1993) later combined light tracing and path tracing into Bidirectional Path Tracing (BDPT), and Veach (1997) introduced provably optimal weighting strategies to maximize convergence speed. Keller (1995) and Owen (1995) improved convergence of Monte-Carlo samplers in general by using low discrepancy samples. Furthermore, Veach and Guibas (1997) proposed a new mutation-based sampling strategy which exceeds in finding low-probability high-contribution paths which can cause significant noise in classic Monte Carlo methods. In addition, Manifold Exploration techniques were developed which excel at discovering highly specular paths (Jakob and S. Marschner, 2012).

Photon Mapping Simultaneously, a number of generally biased radiance estimators emerged that are based on storing outgoing flux *(photons)* in spatial data structures (original work: Jensen, 1996; overview: Kang et al., 2016). This method is conceptually equivalent to Arvo's light tracing approach (Arvo et al., 1986) but stores lighting information in space opposed to on surfaces. The original estimator of Jensen (1996) was made consistent by Hachisuka, Ogaki, et al. (2008) and Knaus and Zwicker (2011) and extended by Hachisuka and Jensen (2009) to also handle glossy surfaces efficiently. Georgiev et al. (2012) further improved the quality by integrating bidirectional path tracing and shadow tests into photon mapping.

Real-time Photon Mapping Jensen (1996) originally proposed to use k-d-trees to perform fast Fixed-Radius-Near-Neighbor (FRNN) queries. Yet, k-d-tree construction and querying are highly incoherent tasks that are commonly implemented using stacks. GPU hardware neither natively supports stacks nor extensive branching, making k-d-trees impractical for real-time rendering. To solve this, Hachisuka and Jensen (2010) proposed to stochastically store photons in a fixed-size hash grid, which improves parallelization and reduces memory access, yet increases noise. Mara et al. (2013) later published four further GPU-optimized photon mapping algorithms, two of which rasterize photons and two use grids. The problem with such linear data structures nonetheless is that they either require stochastic sampling, increasing variance, or linear counting, increasing runtime. With the recent advancements in GPU architectures however, traversal of bounding volume hierarchies has become hardware accelerated. Evangelou et al. (2021) successfully leverage these modern hardware features to perform fast FRNN queries, leading Kern et al. (2023) to apply this to photon mapping and together with culling and stochastic rejection they achieve real-time performance.

Path Guiding Alternatively, to obtain unbiased estimators, photon maps (Jensen, 1996) can also be used only to *guide* path sampling (Jensen, 1995). Vorba et al. (2014) later improved upon this idea by learning Gaussian Mixture Models. Müller, Gross, et al. (2017) instead store an approximation of the light field in a tree structure which is more efficient to construct.

2.2 Path Reuse

Precomputed GI Under the assumption of static geometry and lighting, radiance information can also be precomputed and reused at runtime. The first paper to explicitly introduce this concept was published by Greger et al. (1998) who proposed precomputing irradiance inside a volume to capture indirect illumination, although photon maps (Jensen, 1996) and illumination maps (Arvo et al., 1986; Heckbert, 1990) were already similar in concept. Sloan et al. (2002) expanded on this idea by using spherical harmonics to represent the radiance field, allowing for real-time directional global illumination by simply evaluating the dot product between the coefficients of the radiance field and the surface transfer function. A more approachable introduction to their paper is given by Green (2003). Since their invention, various improvements to the original

method were made (Křivánek, Gautron, G. Ward, et al., 2007). Notably, Kautz et al. (2002) generalized this technique to arbitrary BRDFs. McGuire et al. (2017) approached the light leaking problem by introducing light probes that capture radial distance together with radiance in octahedral maps (Cigolle et al., 2014; Engelhardt and Dachsbacher, 2008) which they use for efficient visibility tests. These precomputed methods have become ubiquitous in real-time rendering applications (Iwanicki and Sloan, 2017; O'Donnell, 2018) but have the fundamental drawback of being static.

Dynamic Radiance Caching In contrast to precomputed GI techniques, radiance caching algorithms are fully dynamic and online. G. J. Ward et al. (1988) lay the groundwork for radiance caching with their seminal paper on the topic. Their key idea was to evaluate the radiance field only sparsely and to render full resolution images by interpolating nearby samples. Tole et al. (2002) generalized this approach to dynamic scenes and arbitrary radiance estimators and Křivánek, Gautron, Pattanaik, et al. (2005) used spherical harmonics to enable radiance caching for generic BSDFs.

Recently, dynamic radiance caching has received renewed attention with several game engines and companies shipping it in production for dynamic GI. Majercik, Guertin, et al. (2019) use dynamic probe placement to make the light probing technique of McGuire et al. (2017) dynamic, which was used in major game titles at Ubisoft (Kuenlin, 2024).

Wright (2021) and Wright et al. (2022) combine screen- and world-space radiance caching with efficient screen-space ray tracing and SDF cone tracing to achieve real-time global illumination in production in Unreal Engine 5. Building on this, Boissé et al. (2023) replaced the world-space cache with a spatial hash and Tatzgern et al. (2024) explored the use of On-Surface Caches.

At EA, Apers (2024), Halén et al. (2021), and Stachowiak (2018) place a fixed amount of surface elements *(surfels)* on the scene geometry on the fly to accumulate irradiance. These surfels are also used for self-learning to simulate infinite bounce indirect lighting.

Path Space Filtering Another interesting and closely related technique is that of Path Space Filtering (Keller et al., 2016), where similar path segments are averaged to reduce noise. Binder et al. (2021) adapted this technique to GPU hardware by introducing spatial hash grids.

Reservoir-based Resampling (ReSTIR) The most recent advancements in path tracing are based on spatio-temporal path reuse through reservoir resampling (original work: Bitterli et al., 2020; generalization: Lin et al., 2022; see also Wyman et al., 2023) and already achieve acceptable noise at close to real-time performance. The core idea is to draw samples from a pool of reused candidates according to their estimated contribution, and was already motivated by Talbot (2005). These methods are also fully dynamic, yet they differ from radiance caching in that they are theoretically unbiased. Although the original paper used reservoir resampling

only for direct illumination (Bitterli et al., 2020), the idea was extended among others to indirect lighting (Ouyang et al., 2021), full path tracing (Lin et al., 2022), photon mapping (Kern et al., 2024) and bidirectional path tracing (Hedstrom et al., 2025). ReSTIR can also be combined with other techniques like radiance caching (Boissé et al., 2023; Majercik, Müller, et al., 2021; Müller, Rousselle, et al., 2021).

2.3 Neural Rendering

With the recent advancements in Deep Learning, neural rendering techniques have gained great interest in the rendering community and have been applied to a vast array of rendering problems.

Neural GI Several papers exist that add lighting effects as a screen-space post-processing effect to deferred shading buffers. Notably, Nalbach et al. (2017) use a convolutional U-net that was trained on a wide variety of scenes, while Thomas and Forbes (2018) use a generative adversarial network (GAN) trained on a per-scene basis. However, in contrast to world-space methods, screen-space methods have only a limited amount of information about the scene available. Hermosilla et al. (2019) thus trained a network to predict global illumination effects on point clouds. Other works use per-scene trained networks to approximate indirect illumination in world space (Ren et al., 2013).

Neural Path Guiding Müller, McWilliams, et al. (2019) use a specialized variant of normalized flow networks to learn a probability distribution over the light field, which they use for importance sampling. Building on this, Müller, Rousselle, et al. (2020) were able to further reduce variance by learning Neural Control Variates.

Neural Interpolation Iwanicki and Sloan (2017) use a neural network to efficiently weight irradiance probes in space to prevent light leaking. Similarly, Zhu et al. (2020) use a neural network to predict kernels for photon mapping based on the neighborhood of photons to improve the quality of density estimation.

Neural Radiance Caching Neural Radiance Fields (NeRF) (Mildenhall et al., 2020) have recently received a lot of attention in the field of Computer Vision to represent 3D scenes. Müller, Rousselle, et al. (2021) used a NeRF-like neural network architecture as backing storage for radiance caching, which they call Neural Radiance Caching (NRC). An interesting idea which they implemented in their paper was given by Dahm and Keller (2017), who propose to understand learning of light fields as reinforcement learning and suggest self-learning for faster adaptation. Müller, Evans, et al. (2022) improved on the original NRC-paper by introducing a versatile positional input encoding based on feature vectors distributed in multi-resolution hash grids. Recently, Dereviannykh et al. (2024) adapted the NRC to cache *incident* radiance. Furthermore, they removed the bias of their method by integrating NRC together with traditional

path tracing in a Two Level Monte Carlo estimator.

Neural Denoising Because Monte-Carlo rendering algorithms are unbiased but noisy, denoising can be highly effective to trade the remaining variance for a small bias. Although hand-crafted spatio-temporal filters can also achieve good results (Schied et al., 2017), recurrent autoencoders are well-suited for high quality screen-space denoising of Monte-Carlo rendered images (Chaitanya et al., 2017) and are nowadays widely used in production (Áfra, 2019). Similarly, Jiang and Kainz (2020) use a convolutional autoencoder to denoise low sample count light probes to predict indirect illumination. For interactive applications, Hasselgren et al. (2020) combine temporally stable denoising with adaptive sampling. On a further note, Neural Radiance Caching (Müller, Rousselle, et al., 2021) can also be thought of as a form of Path Space Filtering (Keller et al., 2016).

2.4 Prerequisites

This thesis mainly builds on the NRC by Müller, Rousselle, et al. (2021) with the extension of Müller, Evans, et al. (2022), but I will introduce a decoupling of the radiance cache similar to what Tole et al. (2002) and Walter, Drettakis, et al. (1999) did to combine it with bidirectional path tracing (Veach, 1997), light tracing (Arvo et al., 1986) and hardware accelerated progressive photon mapping (Hachisuka, Ogaki, et al., 2008; Jensen, 1996; Kern et al., 2023).

3 Constructing a Reference Pathtracer

3.1 Solving the Rendering Equation

As always in rendering, we are interested in solving the rendering equation, which describes how light interacts with surfaces in a scene and was first introduced by Kajiya (1986). The rendering equation is given by:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}_o) = L_e(\boldsymbol{x}, \boldsymbol{\omega}_o) + \int_\Omega f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) L_i(\boldsymbol{x}, \boldsymbol{\omega}_i) \cos\theta_i \, d\boldsymbol{\omega}_i \tag{3.1}$$

where L_o is the outgoing radiance, L_e is the emitted radiance, f is the bidirectional scattering distribution function (BSDF), L_i is the incoming radiance from direction ω_i, and θ_i is the incident angle, which is the angle between the incoming light direction ω_i and the surface normal at $\boldsymbol{x}$.

3.2 Monte Carlo Integration

The rendering equation is an infinite recursive integral over all possible light paths, for which no closed-form solution exists in general. Hence, to solve it we need to resort to numerical integration strategies. Deterministic integration methods suffer from the curse of dimensionality as the rendering equation is notoriously high dimensional. Thus, we utilize probabilistic approaches to solve it. We will use Monte Carlo integration, which was originally developed by Metropolis and Ulam (1949) as part of the Manhattan Project and first introduced to rendering by Kajiya (1986). The idea is, to compute the integral using random samples drawn from a probability distribution, which ideally is chosen as proportional to the integrand as possible.

Applying this to the rendering equation, we can formulate an estimator for the outgoing radiance L_o:

$$\langle I_{\mathrm{MC}} \rangle_N = L_e(\boldsymbol{x}, \boldsymbol{\omega}_o) + \frac{1}{N} \sum_{i=1}^{N} \frac{f(\hat{\boldsymbol{\omega}}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) \cos(\hat{\theta}_i)}{p(\hat{\boldsymbol{\omega}}_i \mid \boldsymbol{\omega}_o)} L_i(\boldsymbol{x}, \hat{\boldsymbol{\omega}}_i), \quad \hat{\boldsymbol{\omega}}_i \sim p(\hat{\boldsymbol{\omega}}_i \mid \boldsymbol{\omega}_o) \tag{3.2}$$

This is an unbiased estimator for the rendering equation, as can be easily shown:

$$\begin{aligned}\mathbb{E}[\langle I_{\mathrm{MC}}\rangle_N] &= L_e(\boldsymbol{x},\boldsymbol{\omega}_o) + \frac{1}{\cancel{N}}\sum_{i=1}^{\cancel{N}}\mathbb{E}_{\hat{\boldsymbol{\omega}}_i\sim p(\hat{\boldsymbol{\omega}}_i|\boldsymbol{\omega}_o)}\left[\frac{f(\hat{\boldsymbol{\omega}}_i,\boldsymbol{x},\boldsymbol{\omega}_o)\cos(\hat{\theta}_i)}{p(\hat{\boldsymbol{\omega}}_i \mid \boldsymbol{\omega}_o)}L_i(\boldsymbol{x},\hat{\boldsymbol{\omega}}_i)\right] \\ &= L_e(\boldsymbol{x},\boldsymbol{\omega}_o) + \int_\Omega \frac{f(\hat{\boldsymbol{\omega}}_i,\boldsymbol{x},\boldsymbol{\omega}_o)\cos(\hat{\theta}_i)\cancel{p(\hat{\boldsymbol{\omega}}_i \mid \boldsymbol{\omega}_o)}}{\cancel{p(\hat{\boldsymbol{\omega}}_i \mid \boldsymbol{\omega}_o)}}L_i(\boldsymbol{x},\hat{\boldsymbol{\omega}}_i)\,d\boldsymbol{\omega}_i \\ &= L_o(\boldsymbol{x},\boldsymbol{\omega}_o)\end{aligned} \tag{3.3}$$

Hence, by the law of large numbers this estimator converges to the true solution to the rendering equation for $N \to \infty$.

To handle the infinite recursion without introducing bias, I use Russian Roulette Termination as motivated by Veach (1997), which probabilistically terminates the recursion based on the contribution of the current path segment:

$$\langle I_{\mathrm{RR}}\rangle_N = \begin{cases}\frac{\langle I_{\mathrm{MC}}\rangle_N}{p_{\mathrm{continue}}} & \text{with probability } p_{\mathrm{continue}} \\ 0 & \text{with probability } 1 - p_{\mathrm{continue}}\end{cases} \tag{3.4}$$

The probability terms cancel out, thus this modification to the estimator does not introduce bias. I choose p_{continue} proportional to the relative luminance Y (Kirkpatrick et al., 2025) of the contribution of the current path segment, which measures the perceived linear brightness of that sample:

$$Y(\boldsymbol{c}) = 0.2126 \cdot c_r + 0.7152 \cdot c_g + 0.0722 \cdot c_b \tag{3.5}$$

3.3 Randomized Quasi-Monte Carlo Integration

To improve the convergence of the Monte Carlo integration, low-discrepancy sequences can be used instead of purely random samples. Low-discrepancy sequence are designed to fill the integration domain more uniformly than random samples, which can lead to a lower variance in the estimator. This approach is called Quasi-Monte Carlo integration and was first introduced to rendering by Heinrich and Keller (1994). Keller (1996) later showed faster convergence of Quasi-Monte Carlo integration compared to random Monte Carlo integration in exemplary path tracing scenarios.

Carefully introducing randomness to the low-discrepancy sequence can help to improve convergence further and mitigate visible artifacts caused by the deterministic nature of such sequences. This is known as Randomized Quasi-Monte Carlo integration and was first introduced by Owen (1995). For a practical application to path tracing see for example Burley (2020).

I use the scrambled Sobol' sequence from cuRAND (NVIDIA Corporation, 2024) which is based on the work of Owen (2008). Furthermore, I employ the same technique as the Blender

Foundation (2025) and only precompute a single Sobol' sequence for the entire scene, which is then decorrelated between pixels using constant random per-pixel shifts modulo 1, also known as Cranley-Patterson rotations (Cranley and Patterson, 1976). The Sobol' sequence is regularly recomputed in batches on the GPU to allow for unlimited progressive rendering, the per pixel shifts are only recomputed on frame buffer resize, which makes this technique relatively lightweight.

3.4 Principled BSDF

As a material model we will use a simplified physically based model, which is inspired by the Disney principled BSDF (Burley and Walt Disney Animation Studios, 2012) and conformant with the glTF PBR specification (The Khronos® 3D Formats Working Group, 2021). The BSDF will be restricted to the diffuse, specular and transmission components, as they can already cover a wide range of interesting materials and suffice to showcase the strengths and weaknesses of the subsequent techniques.

The BSDF is split into two parts: If the light direction $\boldsymbol{\omega}_i$ and the view direction $\boldsymbol{\omega}_o$ lie in the same hemisphere, the BSDF is composed of a diffuse and a specular lobe, otherwise it only consists of a transmission lobe:

$$f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) = \begin{cases} f_s(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) + f_d(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) & \text{if } \langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_i \,\rangle \cdot \langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_o \,\rangle > 0 \\ f_t(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) & \text{otherwise} \end{cases} \tag{3.6}$$

For the specular component, we will use the Torrance-Sparrow/Cook-Torrance microfacet model (Cook and Torrance, 1982; Torrance and Sparrow, 1967) with the corrected normalization factor from Walter, S. R. Marschner, et al. (2007):

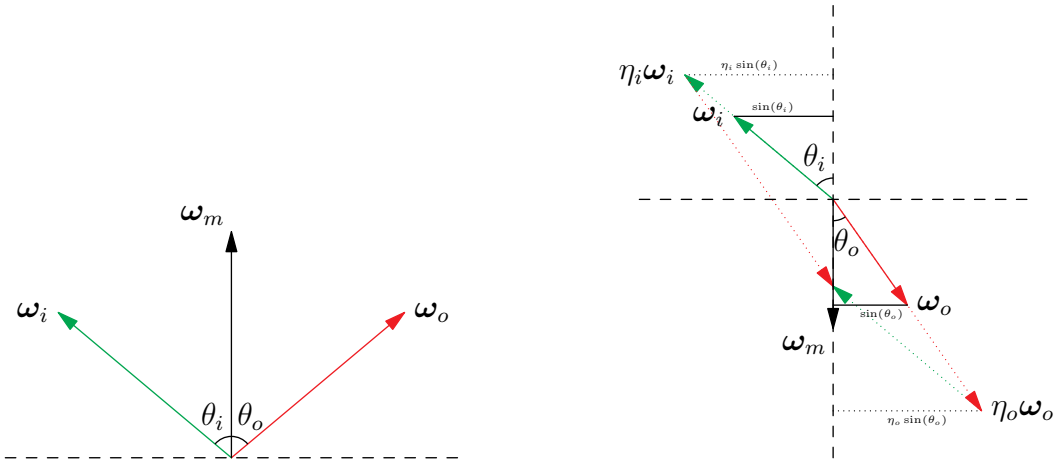
$$f_s(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) = \frac{D(\langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_m \,\rangle) G(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) F(\langle\, \boldsymbol{\omega}_m \,|\, \boldsymbol{\omega}_i \,\rangle)}{4 \langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_i \,\rangle \langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_o \,\rangle} \tag{3.7}$$

where D is the isotropic Trowbridge-Reitz normal distribution function (NDF) originally defined by Trowbridge and Reitz (1975) and later rediscovered by Walter, S. R. Marschner, et al. (2007):

$$D(\langle\, \boldsymbol{n} \,|\, \boldsymbol{\omega}_m \,\rangle) = \frac{\alpha^2}{\pi(\alpha^2 \cos^2 \theta_m + \sin^2 \theta_m)^2} \tag{3.8}$$

Here, α is the width of the microfacet distribution, which is related to the roughness of the surface. $\boldsymbol{\omega}_m$ is the normal of an ideally reflecting microfacet (see figure 3.1a):

$$\boldsymbol{\omega}_m = \frac{\boldsymbol{\omega}_i + \boldsymbol{\omega}_o}{\|\boldsymbol{\omega}_i + \boldsymbol{\omega}_o\|} \tag{3.9}$$



(a) Reflection (b) Refraction

Figure 3.1: **(a)** For the reflective case, the halfvector ω_m lies simply in the middle of the incident and outgoing direction. **(b)** We can obtain the halfvector ω_m for the refractive case by summing $\eta_i\omega_i$ and $\eta_o\omega_o$, where η_i and η_o are the indices of refraction of the incident and outgoing medium, respectively. The orthogonal components of ω_i and ω_o are simply the sine, so by weighting and summing them they cancel out according to Snell's law.

G is the corresponding geometric attenuation term, for which I use the Smith shadowing-masking function (Smith, 1967), derived for the Trowbridge-Reitz NDF by Walter, S. R. Marschner, et al. (2007):

$$\begin{aligned} G(\boldsymbol{n}, \omega_i, \omega_o) &= \frac{1}{1 + \Lambda(\langle\, \boldsymbol{n} \,|\, \omega_i \,\rangle) + \Lambda(\langle\, \boldsymbol{n} \,|\, \omega_o \,\rangle)} \\ \text{with} \quad \Lambda(\langle\, \boldsymbol{n} \,|\, \omega \,\rangle) &= \frac{1}{2}\left(\sqrt{1 + \alpha^2 \tan^2\theta} - 1\right) \end{aligned} \tag{3.10}$$

For the Fresnel term F, I use the Schlick approximation (Schlick, 1994):

$$F(\langle\, \omega_m \,|\, \omega_i \,\rangle) = F_0 + (1 - F_0)(1 - \langle\, \omega_m \,|\, \omega_i \,\rangle)^5 \tag{3.11}$$

In the case of transmission, the normal of an ideally refracting microfacet is given by (see figure 3.1b):

$$\omega_m = \pm\frac{\eta_i\omega_i + \eta_o\omega_o}{\|\eta_i\omega_i + \eta_o\omega_o\|} = \pm\frac{\eta\omega_i + \omega_o}{\|\eta\omega_i + \omega_o\|} \tag{3.12}$$

where $\eta = \frac{\eta_i}{\eta_o}$ is the relative index of refraction (IOR). The transmission component is then the following, as derived by Walter, S. R. Marschner, et al. (2007):

$$f_t(\omega_i, \boldsymbol{x}, \omega_o) = \rho_t \frac{D(\langle\, \boldsymbol{n} \,|\, \omega_m \,\rangle) G(\boldsymbol{n}, \omega_i, \omega_o)(1 - F(\langle\, \omega_m \,|\, \omega_i \,\rangle))}{(\langle\, \omega_m \,|\, \omega_i \,\rangle + \langle\, \omega_m \,|\, \omega_o \,\rangle/\eta)^2} \left| \frac{\langle\, \omega_m \,|\, \omega_i \,\rangle \langle\, \omega_m \,|\, \omega_o \,\rangle}{\langle\, \boldsymbol{n} \,|\, \omega_i \,\rangle \langle\, \boldsymbol{n} \,|\, \omega_o \,\rangle} \right| \tag{3.13}$$

For the diffuse component, we simply use the Lambertian reflectance model:

$$f_d(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) = \frac{\rho_d}{\pi} \tag{3.14}$$

Several other, more physically accurate, models have been proposed for the diffuse component, such as the microfacet-based Oren-Nayar reflectance model (Oren and Nayar, 1994) or the heuristic model from Burley and Walt Disney Animation Studios (2012). The Oren-Nayar model is rather computational expensive and not fully energy conserving, although these deficiencies can be mostly overcome by the improvements made by Fujii (2025). Burley's model also suffers from energy conservation issues, but renormalized versions exist (Lagarde and de Rousiers, 2014). However, the Lambertian model has the nice property of being constant over the hemisphere, which can be advantageous for caching.

To parameterize the BSDF, we will use a base color $\boldsymbol{c} \in [0, 1]^3$, a metallic factor $m \in [0, 1]$, a roughness factor $r \in [0, 1]$ and a transmission factor $t \in [0, 1]$, as follows:

$$\begin{aligned} \alpha &= r^2 \\ \rho_d &= (1 - m) \cdot (1 - t) \cdot \boldsymbol{c} \\ \rho_t &= (1 - m) \cdot t \cdot \boldsymbol{c} \\ F_0 &= 0.04 \cdot (1 - m) + \boldsymbol{c} \cdot m \end{aligned} \tag{3.15}$$

Here, we assume the IOR of the surface to be 1.5, which results in a specular reflectance at normal incidence of $F_0 = 0.04$. For metals, the base color $\boldsymbol{c}$ is used directly as the F_0 to simulate wavelength dependent complex-valued IOR.

3.5 Importance Sampling

As mentioned above, the variance of a Monte Carlo estimator can be reduced by sampling the integral using a probability distribution that is as proportional to the integrand as possible. Revisiting the rendering equation (equation (3.1)), we first concentrate on sampling the term $f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) \cos(\theta_i)$ of the integrand. Also sampling $L_i(\boldsymbol{x}, \boldsymbol{\omega}_i)$ is not that easy, because to obtain L_i we would need to solve the rendering equation again. Yet, several approaches have been made to also take L_i into account, notably direct light sampling (Veach, 1997), path guiding (Jensen, 1995; Lafortune and Willems, 1995; Müller, Gross, et al., 2017; Müller, McWilliams, et al., 2019) and ReSTIR (Bitterli et al., 2020). For my reference pathtracer, I will only employ BSDF sampling and direct light sampling (section 3.7).

For the diffuse component, as it is constant I only sample the cosine weighted hemisphere:

$$p_d(\boldsymbol{\omega}_i) = \frac{\cos(\theta_i)}{\pi} \quad \text{for } \boldsymbol{\omega}_i \in \Omega \tag{3.16}$$

For the specular and transmissive components, I sample the distribution of visible microfacet normals (VNDF) (Heitz, 2014) of the Trowbridge-Reitz NDF, which is given by:

$$p_{\text{VNDF}}(\omega_m \mid \omega_o) = \frac{G_1(\omega_o, \omega_m) D(\langle \boldsymbol{n} \mid \omega_m \rangle) |\langle \omega_m \mid \omega_o \rangle|}{|\langle \boldsymbol{n} \mid \omega_o \rangle|} \tag{3.17}$$

Sampling this instead of the NDF reduces the variance at grazing angles, where many microfacets are shadowed. To sample the VNDF, I use the method proposed by Dupuy and Benyoub (2023), which improves upon the original method by Heitz (2014) and its follow-up (Heitz, 2018).

To weigh the BSDF, the VNDF has first to be transformed into a probability distribution over the domain of incoming directions ω_i using the Jacobian of the reflection operator (Walter, S. R. Marschner, et al., 2007):

$$\begin{aligned} \omega_i &= \text{reflect}(\omega_o, \omega_m), \quad \omega_m \sim p_{\text{VNDF}}(\omega_m \mid \omega_o) \\ p_s(\omega_i \mid \omega_o) &= p_{\text{VNDF}}(\omega_m \mid \omega_o) \left\| \frac{\partial \omega_m}{\partial \omega_i} \right\| = \frac{p_{\text{VNDF}}(\omega_m \mid \omega_o)}{4|\langle \omega_m \mid \omega_o \rangle|} = \frac{G_1(\omega_o, \omega_m) D(\langle \boldsymbol{n} \mid \omega_m \rangle)}{4|\langle \boldsymbol{n} \mid \omega_o \rangle|} \end{aligned} \tag{3.18}$$

$\left\| \frac{\partial \omega_m}{\partial \omega_i} \right\|$ denotes the absolute value of the determinant of the Jacobian for the transform from ω_m to ω_i and was derived by Walter, S. R. Marschner, et al. (2007).

For the transmission component, I use the same approach, but with the Jacobian of the refraction operator (Walter, S. R. Marschner, et al., 2007):

$$\begin{aligned} \omega_i &= \text{refract}(\omega_o, \omega_m), \quad \omega_m \sim p_{\text{VNDF}}(\omega_m \mid \omega_o) \\ p_t(\omega_i \mid \omega_o) &= p_{\text{VNDF}}(\omega_m \mid \omega_o) \left\| \frac{\partial \omega_m}{\partial \omega_i} \right\| \\ &= p_{\text{VNDF}}(\omega_m \mid \omega_o) \frac{\eta_i^2 |\langle \omega_m \mid \omega_i \rangle|}{|\eta_i \langle \omega_m \mid \omega_i \rangle + \eta_o \langle \omega_m \mid \omega_o \rangle|^2} \\ &= \frac{G_1(\omega_o, \omega_m) D(\langle \boldsymbol{n} \mid \omega_m \rangle) |\langle \omega_m \mid \omega_o \rangle|}{|\langle \boldsymbol{n} \mid \omega_o \rangle|} \frac{|\langle \omega_m \mid \omega_i \rangle|}{|\langle \omega_m \mid \omega_i \rangle + \langle \omega_m \mid \omega_o \rangle / \eta|^2} \\ &= \frac{G_1(\omega_o, \omega_m) D(\langle \boldsymbol{n} \mid \omega_m \rangle) |\langle \omega_m \mid \omega_i \rangle \langle \omega_m \mid \omega_o \rangle|}{|\langle \boldsymbol{n} \mid \omega_o \rangle| |\langle \omega_m \mid \omega_i \rangle + \langle \omega_m \mid \omega_o \rangle / \eta|^2} \end{aligned} \tag{3.19}$$

To combine these three sampling strategies, one can use Russian Roulette again. To optimally weigh the sampling strategies, one can first precompute the relative luminance Y of the directional albedo of the respective BSDF components, which is the expected value of the BSDF in a perfect white furnace environment. However, for performance reasons, I only approximate these weights

by using the Fresnel term of a perfectly smooth surface and the base colors:

$$\begin{aligned} Y_d &= Y(\rho_d) & Y_s &= Y(F(\langle \boldsymbol{n} \mid \boldsymbol{\omega}_o \rangle)) & Y_t &= Y(\rho_t) \\ P_d &= \frac{Y_d}{Y_d + Y_s + Y_t} & P_s &= \frac{Y_s}{Y_d + Y_s + Y_t} & P_t &= \frac{Y_t}{Y_d + Y_s + Y_t} \end{aligned} \tag{3.20}$$

The combined sampling distribution is then given by:

$$p(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o) = P_d p_d(\boldsymbol{\omega}_i) + P_s p_s(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o) + P_t p_t(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o) \tag{3.21}$$

3.6 Evaluation of the Weighted BSDF

Most of the time, we do not need to evaluate the BSDF directly, but rather the weighted BSDF, which is given by:

$$\begin{aligned} w(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) &= \frac{f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o)\cos(\theta_i)}{p(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)} \\ &= \begin{cases} \frac{(f_d(\boldsymbol{\omega}_i,\boldsymbol{x},\boldsymbol{\omega}_o)+f_s(\boldsymbol{\omega}_i,\boldsymbol{x},\boldsymbol{\omega}_o))\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle}{P_d p_d(\boldsymbol{\omega}_i)+P_s p_s(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)} & \text{if } \langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle \langle \boldsymbol{n} \mid \boldsymbol{\omega}_o \rangle > 0 \\ \frac{f_t(\boldsymbol{\omega}_i,\boldsymbol{x},\boldsymbol{\omega}_o)\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle}{P_t p_t(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)} & \text{otherwise} \end{cases} \end{aligned} \tag{3.22}$$

Using this formulation is more efficient because large parts cancel out, improving numerical stability and performance. For the reflective part we then get:

$$\begin{aligned} w_r(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) &= \frac{(f_d(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) + f_s(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o))\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle}{P_d p_d(\boldsymbol{\omega}_i) + P_s p_s(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)} = \frac{G_1^{-1}(kG^{-1}\rho_d + \pi DF)}{G^{-1}(kG_1^{-1}P_d + \pi DP_s)}, \\ k &= 4\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle \langle \boldsymbol{n} \mid \boldsymbol{\omega}_o \rangle \end{aligned} \tag{3.23}$$

And for the transmissive part:

$$w_t(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) = \frac{f_t(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o)\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle}{P_t p_t(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)} = \rho_t \frac{G(1-F)}{G_1 P_t} = \rho_t \frac{G_1^{-1}(1-F)}{G^{-1}P_t} \tag{3.24}$$

I used the fact that G and G_1 are both reciprocals to eliminate unnecessary divisions:

$$G^{-1} = 1 + \Lambda(\langle \boldsymbol{n} \mid \boldsymbol{\omega}_i \rangle) + \Lambda(\langle \boldsymbol{n} \mid \boldsymbol{\omega}_o \rangle) \quad \text{and} \quad G_1^{-1} = 1 + \Lambda(\langle \boldsymbol{n} \mid \boldsymbol{\omega}_o \rangle) \tag{3.25}$$

3.7 Multiple Importance Sampling

BSDF importance sampling already works well for glossy materials and diffuse lighting. When it comes to diffuse materials and direct lighting however, it has rather high variance and does not

work at all for infinitesimal light sources. In these scenarios, direct light sampling is much more efficient, which samples the light sources instead of the BSDF.

To maximize path reuse, a common strategy is to draw from both sampling strategies at every path vertex, using the BSDF sampling for path continuation and the light sampling for light collection. This technique is called Next Event Estimation (NEE).

A general approach to combine multiple sampling strategies is given by the Multiple Importance Sampling (MIS) estimator, which was first introduced by Veach (1997):

$$\langle I_{\text{MIS}} \rangle_N = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(\hat{x}_{i,j}) \frac{f(\hat{x}_{i,j})}{p_i(\hat{x}_{i,j})}, \quad \hat{x}_{i,j} \sim p_i(\hat{x}_{i,j}) \tag{3.26}$$

where N is the number of sampling strategies, N_i is the number of samples drawn from the i-th sampling strategy, p_i is the probability distribution of the i-th sampling strategy and $w_i(\hat{x})$ is the weight of the sample $\hat{x}$ drawn from the i-th sampling strategy with $\sum w_i(\hat{x}) = 1$. This estimator is again trivially unbiased.

A naïve way to combine these two sampling strategies would be to simply weigh them equally. However, this is suboptimal, as the resulting variance would then simply be the average of the variances of both sampling strategies. Instead, Veach (1997) also introduced a provably optimal way to combine multiple sampling strategies, called the balance heuristic:

$$w_i(\hat{x}) = \frac{N_i p_i(\hat{x})}{\sum_j N_j p_j(\hat{x})} \tag{3.27}$$

Intuitively spoken, each sampling strategy is weighted according to how likely it is to produce the given sample. Thus, a sample is weighted lower, if it is in an outlier of the generating distribution, and higher, if it is an inlier.

As suggested by Veach (1997), I use the power heuristic generalization with a sharpening exponent of $\beta = 2$ to combine BSDF and light sampling:

$$w_i^{\beta}(\hat{x}) = \frac{(N_i p_i(\hat{x}))^{\beta}}{\sum_j (N_j p_j(\hat{x}))^{\beta}} \tag{3.28}$$

4 Neural Radiance Caching

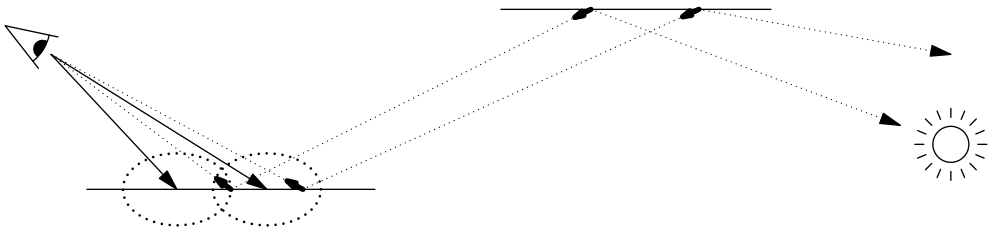


Figure 4.1: Visualization of radiance caching. Full paths (dotted arrows) are only evaluated sparsely, and their radiance estimates are cached per vertex (small arrows). To render the image, short paths (solid arrows) are traced from the camera to collect cached radiance by interpolation (dotted circles).

A weak point of the reference Monte Carlo pathtracer motivated in chapter 3 is, that we have to start from zero, every time the scene or camera changes. It would make sense to keep samples around that can be reused when temporally and spatially similar paths occur in the evaluation of the global illumination integral. Radiance caching by G. J. Ward et al. (1988) addresses this problem by caching sparse estimates of the radiance field. To generate full resolution images, radiance is interpolated from cached estimates.

A particularly interesting variant of radiance caching is Neural Radiance Caching (NRC) by Müller, Rousselle, et al. (2021). They train a neural network to predict the radiance along training paths generated using path tracing. For inference, they then replace the recursive path tracing integral with a neural estimate of the radiance field when the variance along the path becomes too high. The Neural Radiance Cache can be interpreted as a classical cache that uses gradient descent for temporally smooth cache updates and the generalization capabilities of neural networks to interpolate between cached samples. Furthermore, this approach could also be viewed as a form of path space denoising of the radiance field.

Dereviannykh et al. (2024) recently adapted the NRC to predict *incident* radiance instead. Yet, this is probably not well suited for the caustics' problem this thesis aims to solve, which exhibit high-frequency sparse incident radiance but low-frequency dense outgoing radiance. Thus, this thesis only explores classical NRC, although applying the radiance estimators from chapter 5 to

the Neural Incident Radiance Cache (NIRC) may pose an interesting topic for future research.

4.1 Network Architecture

Because the network has to be trained and evaluated in real-time, Müller, Evans, et al. (2022) and Müller, Rousselle, et al. (2021) propose to use a rather small hardware-accelerated network architecture. They use a fully connected network with 3-5 hidden layers, depending on the input encoding, with 64 neurons and ReLU activation functions each. The output layer has 3 values, one for each color channel. Additionally, they omit biases as they did not observe a measurable benefit and it simplifies the implementation. Because of the shallow architecture, skip connections are not needed to preserve the gradient flow.

To maximize the inference and training performance, Müller, Rousselle, et al. (2021) introduce a fully fused implementation of the network, which fully utilizes shared memory and per-thread registers. For this, the whole network is implemented as a single CUDA kernel, with each block handling a batch of 128 64-dimensional input vectors and keeping the 64×128 activations in shared memory. Each block then consists of 4 warps, computing 16 neuron activations each by applying a 16×64 matrix multiplication and the element-wise activation function. The hardware accelerated 16×16 FP16 matrix multiplication instruction is used when available. With this hardware-optimized implementation, Müller, Rousselle, et al. (2021) achieve a speedup of 5× to 10× compared to the widely used TensorFlow framework (TensorFlow Developers, 2021) on large batch sizes corresponding to HD resolution.

4.2 Input Encodings

Table 4.1 The components of the 64-dimensional input vector to the neural network. $\text{mhe}(\cdot)$ is the Multiresolution Hash Encoding from Müller, Evans, et al. (2022), $\text{ob}(\cdot)$ is the One Blob Encoding from Müller, Rousselle, et al. (2021).

Parameter	Encoding	Dimensions
Position $\boldsymbol{x} \in \mathbb{R}^3$	$\text{mhe}(\boldsymbol{x}) \in \mathbb{R}^{16\times2}$	0–31
Direction $\boldsymbol{\omega}_o \in \mathbb{S}^2$	$\text{ob}(\boldsymbol{\omega}_o) \in \mathbb{R}^{2\times4}$	32–39
Normal $\boldsymbol{n}(\boldsymbol{x}) \in \mathbb{S}^2$	$\text{ob}(\boldsymbol{n}(\boldsymbol{x})) \in \mathbb{R}^{2\times4}$	40–47
Roughness $r(\boldsymbol{x}) \in [0, 1]$	$\text{ob}(r(\boldsymbol{x})) \in \mathbb{R}^4$	48–51
Diffuse reflectance $\boldsymbol{F}_d(\boldsymbol{x}, \boldsymbol{\omega}_o) \in [0, 1]^3$	$\boldsymbol{F}_d(\boldsymbol{x}, \boldsymbol{\omega}_o) \in \mathbb{R}^3$	51–53
Specular reflectance $\boldsymbol{F}_s(\boldsymbol{x}, \boldsymbol{\omega}_o) \in [0, 1]^3$	$\boldsymbol{F}_s(\boldsymbol{x}, \boldsymbol{\omega}_o) \in \mathbb{R}^3$	54–57
Padding $\mathbf{1} \in \mathbb{R}^7$	$\mathbf{1} \in \mathbb{R}^7$	57–63

As shown by Ren et al. (2013), parametrizing the network with the outgoing direction $\boldsymbol{\omega}_o$ and the position $\boldsymbol{x}$ alone is not sufficient for an accurate representation of the radiance field. Instead, it proves beneficial to include the surface normal $\boldsymbol{n}$, surface roughness r and the expected diffuse

and specular reflectance terms $\boldsymbol{F}_d$ and $\boldsymbol{F}_s$ as well. The diffuse reflectance $\boldsymbol{F}_d$ is simply the diffuse albedo for Lambertian surfaces. To compute the specular reflectance $\boldsymbol{F}_s$, we have to compute the integral over the hemisphere of incoming directions in a white furnace environment. Fortunately, this can be efficiently precomputed by generating a lookup table over the roughness and the cosine of the viewing angle that stores both a bias and a scaling factor on the Fresnel reflectance at normal incidence $\boldsymbol{F}_0$ (see figure 4.2). The formula and derivation is given by Karis and Epic Games (2013), who use the specular reflectance to filter environment maps for approximate image-based lighting.

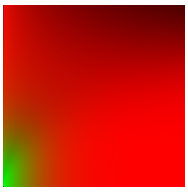


Figure 4.2: Lookup table for the specular reflectance $\boldsymbol{F}_s$. The x-axis is the cosine of the viewing angle, and the y-axis is the roughness. The red channel gives the scaling factor on $\boldsymbol{F}_0$, the green channel the additional constant term.

Furthermore, the correlation between the outgoing radiance and the input features $\boldsymbol{x}$, $\boldsymbol{\omega}_o$, $\boldsymbol{n}$ and r is highly non-linear, so additional effort is required to linearize the input space, as neural networks approximate (roughly) linear functions more efficiently. The diffuse and specular reflectance on the other hand can be used directly, as the outgoing radiance will essentially be a linear combination of these two terms.

4.2.1 Positional Encodings

Triangle Wave Encoding A natural idea to linearize the input space is to find a set of orthonormal basis functions. This enables the neural network to approximate arbitrarily complex functions on the input space as a linear combination of these basis functions. A very famous such basis set is the Fourier series, which forms a basis for the space of square-integrable functions on the interval $[0, 1]$. Square-integrability is not really a constraint, as it essentially assumes finite energy, which is a reasonable assumption to make for the radiance field. Neither is the limitation to $[0, 1]$, as we can always normalize the input to the bounding box of the scene. Such types of encodings were first used in language processing by Vaswani et al. (2017) and later adapted to computer graphics by Tancik et al. (2020) and to neural radiance fields by Mildenhall et al. (2020).

However, for practical applications, compromises are necessary: Firstly, Tancik et al. (2020) make the assumption that the output field is separable in the input dimensions, which is generally not the case, resulting in stripe artifacts in the output if the network is too shallow to adequately mitigate this nonlinearity. Yet, this allows them to apply the Fourier transform individually per

input dimension and brings the number of coefficients down from exponential to linear in the number of dimensions. The hope is, that the neural network can compensate for the missing information. This is generally possible, as every non-separable function can be approximated by a linear combination of separable functions, for example by writing the function as a tensor and applying tensor decomposition. Additionally, Müller, Rousselle, et al. (2021) omit the cosine terms and approximate the sine terms with triangle waves, which they found to reduce the number of coefficients and the computational cost without visibly affecting the output quality.

Instant Neural Graphic Primitives In a follow-up work, Müller, Evans, et al. (2022) propose a new positional encoding scheme, which they call Multiresolution Hash Encoding (MHE). The idea is, to distribute feature vectors in a hierarchical grid in space. For a given input position $\boldsymbol{x}$ and for each grid level a feature vector is then obtained by linearly interpolating the feature vectors at the corners of the current grid cell. The individual feature vectors are concatenated and used as input to the neural network. During training, the gradient is propagated back to the feature vectors. To reduce the size of the backing feature grid and to better represent sparse fields, Müller, Evans, et al. (2022) use a hash table per grid level. For the hyperparametrization of the grid, Müller, Evans, et al. (2022) empirically found 16 levels with 2 features per grid level to be Pareto optimal for their applications, which is adopted here as well.

4.2.2 Directional Encodings

Spherical Harmonics A popular choice for the encoding of direction vectors is the use of spherical harmonics. This is a natural choice, as the spherical harmonics form an orthonormal basis for the space of square-integrable functions defined on the unit sphere, so every function on directions can be represented as a linear combination of spherical harmonics, whose coefficients can be learned by the neural network. However, the number of basis functions is quadratic in the degree m, which limits us to a maximum degree of $m = 3$ for the 64-dimensional input vector, and they are relatively expensive to compute. The limited degree also hinders the representation of high-frequency details in the radiance field. Thus, it is worth-while to also explore alternative encodings for $\boldsymbol{\omega}_o$ and $\boldsymbol{n}$.

One Blob Encoding For bounded domains, Müller, McWilliams, et al. (2019) propose a simple encoding scheme, which is a continuous generalization of the popular one-hot encoding, they call One Blob Encoding. The idea is, to divide the input space into a fixed number of bins along each dimension. Each input vector is then assumed to be distributed over the bins by a Gaussian distribution and the activation of the bin is the probability of the input vector to fall into that bin. This probability is simply the integral of the PDF over the bin, which can be computed using the cumulative distribution function (CDF):

$$\mathrm{ob}(x)_i = \Pr(b_i \leq x < b_{i+1}) = \Pr(x < b_{i+1}) - \Pr(x < b_i), \tag{4.1}$$

where b_i and b_{i+1} are the boundaries of the i-th bin. To improve performance, Müller, Rousselle, et al. (2021) approximate the Gaussian distribution with a fourth order polynomial (see figure 4.3).

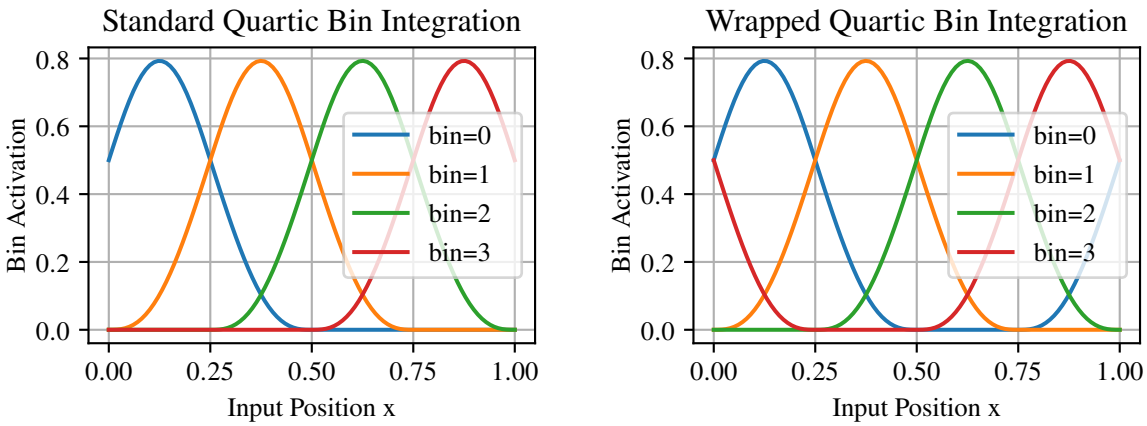


Figure 4.3: Visualization of the activations of the One Blob Encoding for a 1D input space with 4 bins using the quartic approximation of the Gaussian distribution. Left without wrapping at the edges right with.

This encoding is used by Müller, Rousselle, et al. (2021) on the roughness but also on the direction vectors $\boldsymbol{\omega}_o$ and $\boldsymbol{n}$, by first converting these to spherical coordinates and then encoding the azimuthal angle ϕ and the polar angle θ. However, the azimuthal angle is periodic, so the encoding has also to be periodic to avoid discontinuities at the boundaries. To achieve this, Müller, Rousselle, et al. (2021) use a wrapped CDF (see figure 4.3), which comes with the added performance benefit of being able to compute just one boundary per thread and obtaining the other through warp shuffling. Yet, this encoding is not perfect as it establishes singularities at the poles that can become visible at early training stages.

4.2.3 Reflectance Factorization

To help the neural network learn the radiance field, Müller, Rousselle, et al. (2021) propose to multiply the network output with the sum of the mean diffuse and specular reflectance terms:

$$\widehat{L}'(\boldsymbol{x}, \boldsymbol{\omega}_o) = (\boldsymbol{F}_d(\boldsymbol{x}, \boldsymbol{\omega}_o) + \boldsymbol{F}_s(\boldsymbol{x}, \boldsymbol{\omega}_o)) \cdot \mathrm{NRC}(\boldsymbol{x}, \boldsymbol{\omega}_o), \tag{4.2}$$

This way the variance in the training data is reduced, as the remaining part is approximately reduced to just the incoming radiance which is commonly low frequent. Note, however, that these terms are nonetheless given as input, so this information is already available to the network and this is merely a normalization of the training data.



Figure 4.4: Visualization of emission and reflectance factorization. Left: The raw network prediction with emission and reflectance factored out. Right: The prediction multiplied with reflectance and emission added back in.

4.2.4 Emission Factorization

Besides the reflectance factorization, I choose to exclude the emission term from the Neural Radiance Cache and only learn the integral part of the outgoing radiance (see figure 4.4):

$$\widehat{L}(\boldsymbol{x}, \boldsymbol{\omega}_o) = L_e(\boldsymbol{x}, \boldsymbol{\omega}_o) + (\boldsymbol{F}_d(\boldsymbol{x}, \boldsymbol{\omega}_o) + \boldsymbol{F}_s(\boldsymbol{x}, \boldsymbol{\omega}_o)) \cdot \mathrm{NRC}(\boldsymbol{x}, \boldsymbol{\omega}_o), \tag{4.3}$$

This is because the emission term is usually magnitudes higher than the rest of the radiance field and only sparsely present, which leads to high variance in the training data and can lead to leaking of emission into other regions of the cache. In addition, it does not affect performance, as the emission is directly given by the scene geometry, so we can simply add it during inference.

4.3 Training

4.3.1 Collecting Training Data

Algorithm 1 Back transportation of radiance along the training paths

$\boldsymbol{L}_T \leftarrow \boldsymbol{L}_e$
for $i \in (n-1), \ldots, 1$ **do**
  $\boldsymbol{L}_T \leftarrow w_i \cdot \boldsymbol{L}_T$ ▷ Transport radiance to previous vertex
  $\boldsymbol{L}_i \leftarrow \boldsymbol{L}_i + \boldsymbol{L}_T$ ▷ Accumulate radiance at vertex
end for

Müller, Rousselle, et al. (2021) combine the training data collection with the inference step by extending a fraction of the inference paths with additional training bounces to avoid redundant path tracing. However, to enable mixing of different training techniques and to keep flexibility with the implementation, I chose to separate the training data collection from the inference step, accepting potential performance penalties. Yet, Dereviannykh et al. (2024) even found performance benefits in this approach since it decreases incoherence. Nevertheless, the training

sample acquisition is the same: First, trace paths from the camera to the scene, using Russian Roulette termination (equation (3.4)) and Next Event Estimation (section 3.7). At each vertex of the path, store the current contribution weight and the input vector. When sampling an emitter, these weights can then be used to transport the radiance back to the previous vertices (see algorithm 1).

To store the training data, a fixed-size ring buffer with an atomic counter is used to ensure thread safety. This is not ideal, as it may lead to serialization of the GPU threads and thus potentially introduces a performance bottleneck, but it allows for writing dynamic amounts of data per thread. Further research could be done to find a more efficient way to store the training data, for example using warp aggregation and compaction, but this is not straightforward as OptiX does not expose shared memory to allow for Shader Execution Reordering.

4.3.2 Self-Learning

Furthermore, Müller, Rousselle, et al. (2021) propose a self-learning approach, which trades additional bias for reduced variance in the training data. The idea is, to terminate the training paths into the radiance cache from the previous frame to simulate infinite length paths. Because this can amplify the existing bias in the network prediction, they terminate $1/16$ of the training paths in an unbiased manner without self-learning but by Russian Roulette.

4.3.3 Loss Function

Because we use Monte Carlo Sampling to generate the training data, we want the network to learn the mean of the samples in a local region. Furthermore, we assume the samples to be normally distributed by the central limit theorem, so the loss function should be the squared error loss (L2).

However, the network prediction may cover a high dynamic range, especially when the scene contains caustics or highly glossy materials. The traditional L2 loss would thus disproportionately penalize losses in these regions, leading to possible overfitting on bright regions and bad representation of dark regions. To mitigate this, Lehtinen et al. (2018) propose to normalize the L2 loss by the squared prediction of the network. This thesis uses the modification of Müller, Rousselle, et al. (2021), who use the squared *luminance* of the prediction instead for normalization to prevent overfitting onto individual color channels.

4.4 Inference

Using the described training process, the Neural Radiance Cache learns an approximation of the outgoing radiance field, denoted as $\widehat{L}(\boldsymbol{x}, \boldsymbol{\omega}_o)$. For inference, short paths are traced from the camera to the scene using BSDF sampling, the resulting contribution I is then the throughput

along the path multiplied with the radiance estimate at the terminal vertex. The length of the inference paths is a trade-off between bias and variance, as longer paths can cover up cache inaccuracies but introduce Monte Carlo noise. This is analogous to Final Gathering in Photon Mapping (Jensen, 1996). In the following, I will thus discuss different strategies for path termination.

4.4.1 Path Termination Strategies

1st Vertex Terminating directly at the first path vertex is a simple strategy that is well suited to visualize the weaknesses of the radiance cache, but is a rather poor choice for glossy vertices that expose high frequency details in the radiance field.

1st Diffuse Vertex For the aforementioned reasons, it can be beneficial to terminate at the first *diffuse* vertex instead. If a vertex is diffuse can be determined in the simplest case by the luminance of its diffuse albedo ρ_d. This does not include the Fresnel effect however, thus a more sophisticated model could use the directional albedo F_d or compare the importance sampling PDF $p(\boldsymbol{\omega}_i)$ to the Lambertian PDF $p_d(\boldsymbol{\omega}_i)$. Yet, I found this approximation to be sufficient in practice.

Spread Angle Heuristic (SAH) Müller, Rousselle, et al. (2021) propose to terminate paths instead based on the estimated spread of the sampled path. Low spread paths should be path traced, because they exhibit high frequent detail and importance sampling these paths is likely to yield a good estimate. High spread paths on the other hand result in high Monte Carlo noise but low-frequency details, making them suitable for interpolation through radiance caching.

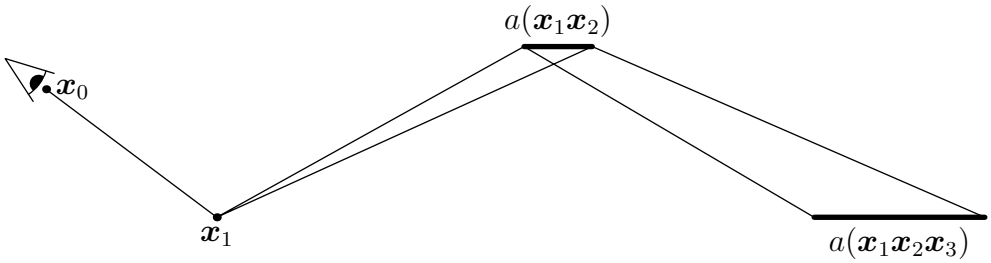


Figure 4.5: The spread along a subpath is proportional to the path length and the inverse of the sampling PDF in solid angle measure, which is intuitively the solid angle covered by the sampled direction.

Müller, Rousselle, et al. (2021) use the findings from Bekaert et al. (2003), according to which the isotropic spread of a subpath $\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$ traced from the camera vertex $\boldsymbol{x}_0$ can be approximated as

$$a(\boldsymbol{x}_1 \cdots \boldsymbol{x}_n) = \left(\sum_{k=2}^{n} \sqrt{\frac{\|\boldsymbol{x}_{k-1} - \boldsymbol{x}_k\|^2}{p(\boldsymbol{\omega}_{ik-1} \mid \boldsymbol{\omega}_{ok-1}) |\cos \theta_{ok}|}} \right)^2, \tag{4.4}$$

where $p(\boldsymbol{\omega}_{ik-1} \mid \boldsymbol{\omega}_{ok-1})$ is the importance sampling PDF of the current vertex in solid angle measure and θ_{ok} is the incident angle at the current vertex. Interestingly, as we will later see, the part under the square root is simply the inverse of the sampling PDF in surface area measure (equation (5.5)), so this term is an estimate for the covered area. By taking the sum over the roots of areas, we are essentially summing the radii. Squaring this sum thus gives us the area spread over the whole path (see figure 4.5).

To terminate paths, they compare the spread of the subpath to the spread at the primary vertex:

$$a(\boldsymbol{x}_1 \cdots \boldsymbol{x}_n) > c \cdot a_0(\boldsymbol{x}_0), \quad a_0(\boldsymbol{x}_0) = \frac{\|\boldsymbol{x}_0 - \boldsymbol{x}_1\|}{4\pi|\cos\theta_{o1}|}, \tag{4.5}$$

where c is a user-defined threshold and 4π is the area of a spherical image plane. Müller, Rousselle, et al. (2021) suggest $c = 0.01$. However, a limitation of this approach is that it will never terminate paths at the first vertex, because the spread is only defined for subpaths with $n \geq 2$.

Balanced Termination Heuristic (BTH) Dereviannykh et al. (2024) propose combining SAH with a stochastic path termination strategy based on the balance heuristic (equation (3.27)) for more aggressive termination already at the first vertex. They define the continuation probability by calculating the MIS weight against a factor $1/\pi$ per NIRC sample. Instead, I propose to use the power heuristic with $\beta = 2$ against the Lambertian diffuse PDF $p_d(\boldsymbol{\omega}_i)$:

$$p_{\text{continue}}(\boldsymbol{\omega}_i) = \frac{p(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)^\beta}{p(\boldsymbol{\omega}_i \mid \boldsymbol{\omega}_o)^\beta + K p_d(\boldsymbol{\omega}_i)^\beta} \tag{4.6}$$

This way the continuation probability will be exactly $1/(K+1)$ if the surface is diffuse, giving an intuitive control over the termination probability.

4.4.2 Temporal Stability

Because the neural network is trained continuously with a relatively high learning rate on highly dynamic data, the output of the network can become temporally unstable. As a solution to this apparent flickering effect, Müller, Rousselle, et al. (2021) propose to use an exponential moving average over the network weights for inference. Note, however, that this is used exclusively for inference, leaving the training process unchanged.

5 Bidirectional Radiance Caching

5.1 A General Theoretical Framework for Radiance Caching

Before proposing new radiance caching techniques, I first want to take a step back and formulate a theoretical backbone. Radiance caching generally can be broken down into four steps:

1. **Training**
   (a) **Query Prediction** First, we want to predict potential queries $\hat{q} = (\boldsymbol{x}, \boldsymbol{\omega}_o)$ according to a distribution $\hat{Q}(\hat{q})$.
   (b) **Radiance Estimation** Given a predicted query $\hat{q}$, we want to estimate the outgoing radiance $L_o(\hat{q})$ at that query. For this, we can use any possible combination of radiance estimation techniques, such as path tracing, light tracing, photon mapping or bidirectional path tracing.
2. **Inference**
   (a) **Query Sampling** We sample queries $q = (\boldsymbol{x}, \boldsymbol{\omega}_o)$ according to a distribution $Q(q)$.
   (b) **Interpolation** Given a query q, we want to approximate the outgoing radiance $\hat{L}_o(q)$ by interpolating the radiance estimates of spatio-temporally nearby query predictions $\hat{q}$. We can use any storing and interpolation technique for this, such as nearest neighbor, linear interpolation or, in our case, the NRC. Note, however, that this step generally introduces *bias*.

Cache Efficiency Using this framework we can observe that the *cache efficiency* is optimal if the query prediction distribution $\hat{Q}(\hat{q})$ is equal to the query sampling distribution $Q(q)$. Conversely, if we sample a query q^* that can not be predicted by the query predictor (i.e. $\hat{Q}(q^*) = 0$), we introduce fundamental bias into the radiance cache $\hat{L}_o(q^*)$, because the radiance cache will not contain an accurate radiance estimate for that query. This happens, whenever the support of the query sampling distribution Q is not contained in the support of the query prediction distribution $\hat{Q}$, i.e. $\text{supp}(Q) \not\subseteq \text{supp}(\hat{Q})$.

In the original formulation of radiance caching by G. J. Ward et al. (1988) and in NRC by Müller, Rousselle, et al. (2021), the filling of the cache and the querying are closely linked. By decoupling these two steps, however, we gain a lot of flexibility in choosing different strategies for both steps. The general idea of decoupling radiance estimation and visualization was already explored by others, notably Walter, Drettakis, et al. (1999) and Tole et al. (2002).

5.2 The Path Space Integral Formulation

To be able to robustly derive the following radiance estimators, we will use an alternative but equivalent formulation of the rendering equation as an integral over the path space, which was introduced by Veach (1997).

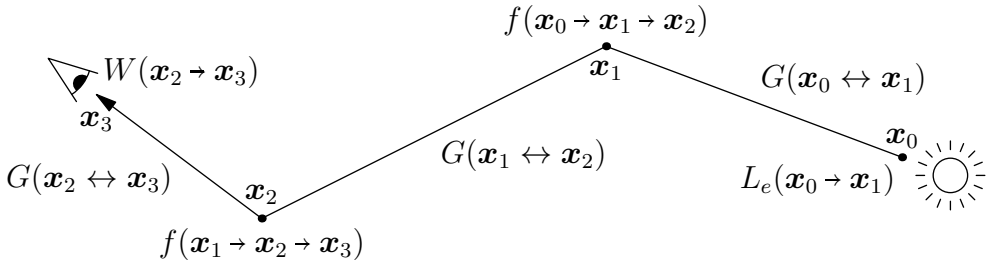


Figure 5.1: Radiance transfer along a path of length $n = 3$ according to the path space integral formulation.

A path $\bar{x}$ of length n is defined as a sequence of vertices $\boldsymbol{x}_0\boldsymbol{x}_1 \cdots \boldsymbol{x}_n$. Let the space of all paths of length n be X_n and the space of all paths $X = \bigcup_{n=1}^{\infty} X_n$. Then, the path space integral formulation of the rendering equation is given by:

$$I = \sum_{n=0}^{\infty} \int_{X_n} L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_1) G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \prod_{i=1}^{n-1} f(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1}) G(\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) \cdot W(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n)\, dA(\boldsymbol{x}_0) \cdots dA(\boldsymbol{x}_n) \tag{5.1}$$

To collect incoming radiance at a sensor point $\boldsymbol{x}_n$, the path space integral formulation contains a sensor weighting term $W(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n)$, which in the case of an infinitesimal pin-hole camera is simply a Dirac-Delta-function over the eye position and a box function over the viewing directions covered by the pixel. The arrow notation $\boldsymbol{x} \to \boldsymbol{y}$ denotes a ray starting at $\boldsymbol{x}$ traveling towards $\boldsymbol{y}$, i.e. $L_o(\boldsymbol{x} \to \boldsymbol{y}) := L_o(\boldsymbol{x}, \boldsymbol{y} - \boldsymbol{x})$. The triples represent surface interactions, $f(\boldsymbol{x} \to \boldsymbol{y} \to \boldsymbol{z}) := f(\boldsymbol{y} - \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} - \boldsymbol{y})$. Double arrow notation is used for path segments, $G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})$ models the radiance transfer between the vertices $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$G(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) := V(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) \frac{\cos\theta_{\boldsymbol{x}\to\boldsymbol{y}} \cos\theta_{\boldsymbol{y}\to\boldsymbol{x}}}{\|\boldsymbol{y} - \boldsymbol{x}\|^2}, \tag{5.2}$$

where $V(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) \in \{0, 1\}$ defines visibility and θ denotes the respective angles of incidence. The integral is measured over the surface of the scene $A(\boldsymbol{x})$.

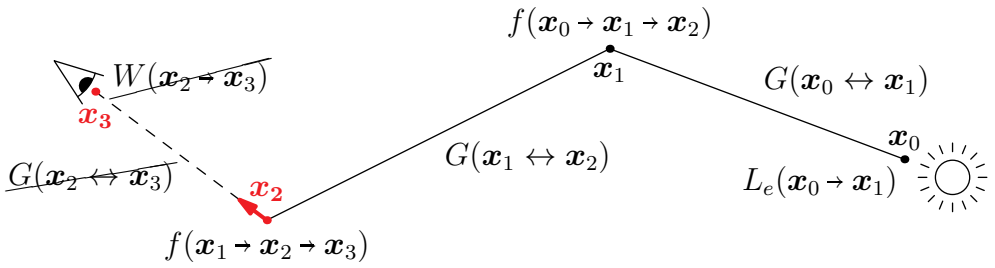


Figure 5.2: To estimate the outgoing radiance $L_o(\boldsymbol{x}_2 \to \boldsymbol{x}_3)$ with the path space integral formulation, we fix the last two vertices defining the position and direction, here $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ highlighted in red. Furthermore, we drop the sensor weighting term W and the last geometry term $G(\boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3)$, because they are not part of the path anymore.

In the following derivations, I will also use a slightly adapted formulation of the path space integral, that estimates *radiance* instead of *intensity* (see figure 5.2). Therefore, we only have to drop the sensor weighting term W and the last geometry term $G(\boldsymbol{x}_{n-1} \leftrightarrow \boldsymbol{x}_n)$ and fix the last two vertices, which define the position and direction of outgoing radiance, by excluding them from the integration domain:

$$\begin{aligned} L_o(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n) = \sum_{n=0}^{\infty} \int_{X_{n-2}} & L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_1) \\ & \cdot \prod_{i=1}^{n-1} G(\boldsymbol{x}_{i-1} \leftrightarrow \boldsymbol{x}_i) f(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1})\, dA(\boldsymbol{x}_0) \ldots dA(\boldsymbol{x}_{n-2}) \end{aligned} \tag{5.3}$$

The main advantage of these formulations is the independence of the individual vertices. This for example allows for straightforward bidirectional sampling.

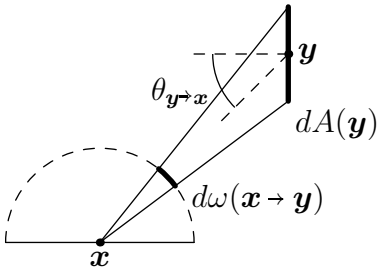


Figure 5.3: Conversion between area and solid angle measure: The differential area $dA(\boldsymbol{y})$ multiplied by the cosine of $\theta_{\boldsymbol{y} \to \boldsymbol{x}}$ is the area apparent from $\boldsymbol{x}$. By dividing by the squared distance it is projected onto the unit sphere, yielding the differential solid angle $d\omega(\boldsymbol{x} \to \boldsymbol{y})$.

An essential tool for the derivation of path samplers is the conversion between area and solid

angle measure (see figure 5.3), given by:

$$d\omega(\boldsymbol{x} \to \boldsymbol{y}) = \frac{\cos\theta_{\boldsymbol{y}\to\boldsymbol{x}}}{\|\boldsymbol{y}-\boldsymbol{x}\|^2}\, dA(\boldsymbol{y}), \tag{5.4}$$

or applied to sampling probabilities:

$$p(\boldsymbol{y})\, dA(\boldsymbol{y}) = p(\boldsymbol{x} \to \boldsymbol{y})\, d\omega(\boldsymbol{x} \to \boldsymbol{y}) \implies p(\boldsymbol{y}) = \left|\frac{d\omega(\boldsymbol{x} \to \boldsymbol{y})}{dA(\boldsymbol{y})}\right| p(\boldsymbol{x} \to \boldsymbol{y}) = \frac{\cos\theta_{\boldsymbol{y}\to\boldsymbol{x}}}{\|\boldsymbol{y}-\boldsymbol{x}\|^2} p(\boldsymbol{x} \to \boldsymbol{y}). \tag{5.5}$$

5.3 Inference

Using this framework a simple inference scheme naturally emerges from the Path Space Integral Formulation (equation (5.1)). The path length n is limited by a path termination strategy (section 4.4.1), so we only need to calculate the finite sum up to the termination length l because we terminate with a valid radiance estimate which incorporates the sum over longer paths. We replace the integral over the path space X_n by a single-sample Monte-Carlo Estimator. For readability, I will do an exemplary derivation for a path of length $n = 2$ without intermediate emission:

$$\begin{aligned}
I &= \int_{X_n} \widehat{L}_o(\boldsymbol{x}_0 \to \boldsymbol{x}_1) G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) f(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2) G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2) \cancel{W(\boldsymbol{x}_1 \to \boldsymbol{x}_2)}\, dA(\boldsymbol{x}_0) \cdots \cancel{dA(\boldsymbol{x}_2)} \\
&= \int_{X_{n-1}} \widehat{L}_o(\boldsymbol{x}_0 \to \boldsymbol{x}_1) G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) f(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2) G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2)\, dA(\boldsymbol{x}_0)\, dA(\boldsymbol{x}_1) && \text{(5.6a)} \\
&\mathrel{\widehat{=}} \frac{\widehat{L}_o(\boldsymbol{x}_0 \to \boldsymbol{x}_1) G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1)}{p(x_0)} \frac{f(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2) G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2)}{p(x_1)} && \text{(5.6b)} \\
&= \frac{\widehat{L}_o(\boldsymbol{x}_0 \to \boldsymbol{x}_1) \cos\theta_{1\to 0}}{p(\boldsymbol{x}_1 \to \boldsymbol{x}_0 \mid \boldsymbol{x}_2 \to \boldsymbol{x}_1)} \frac{f(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2) \cos\theta_{2\to 1}}{p(\boldsymbol{x}_2 \to \boldsymbol{x}_1)} && \text{(5.6c)}
\end{aligned}$$

In the first step, the sensor weighting term W cancels with the integration over the eye vertex (equation (5.6a)). We then replace the integral by a single-sample Monte Carlo estimator (equation (5.6b)). Finally, we convert from area measure to solid angle measure using (equation (5.5)) which cancels with G leaving only the cosine terms at the sampled vertices (equation (5.6c)). The visibility term implicitly vanishes in the ray tracing step.

Applying these observations to all path lengths up to termination, we get:

$$I \mathrel{\widehat{=}} \sum_{i=1}^{n-1} T_i L_e(\boldsymbol{x}_i \to \boldsymbol{x}_{i-1}) + T_n \widehat{L}_o(\boldsymbol{x}_n \to \boldsymbol{x}_{n-1}), \quad T_n = \prod_{i=1}^{n} \frac{f(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1}) \cos\theta_{i\to i+1}}{p(\boldsymbol{x}_i \to \boldsymbol{x}_{i+1} \mid \boldsymbol{x}_{i-1} \to \boldsymbol{x}_i)} \tag{5.7}$$

where T_n is the accumulated throughput along the first n segments of the path and $\widehat{L}_o(\boldsymbol{x}_n \to \boldsymbol{x}_{n-1})$

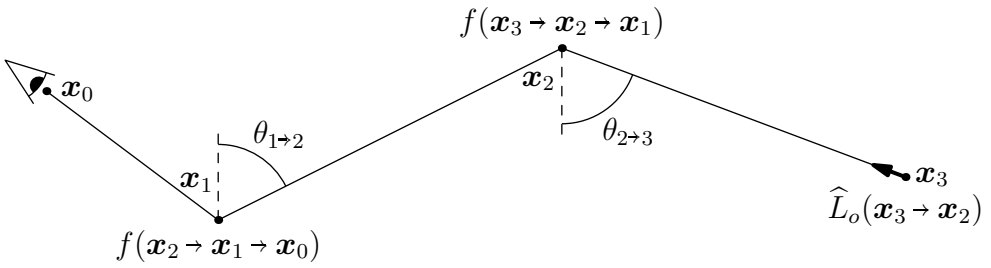


Figure 5.4: Visualization of the inference process.

is the interpolated radiance estimate at the termination vertex. To simplify notation, the path now starts at the eye vertex $\boldsymbol{x}_0$. Note, that we do not have to divide by the path termination probability like we did in Russian Roulette Termination (equation (3.4)), as we terminate the path with a valid radiance estimate.

To improve the quality of the inference pass, we can additionally apply Next Event Estimation with Multiple Importance Sampling like in the following section and the reference pathtracer.

5.4 Path Tracing

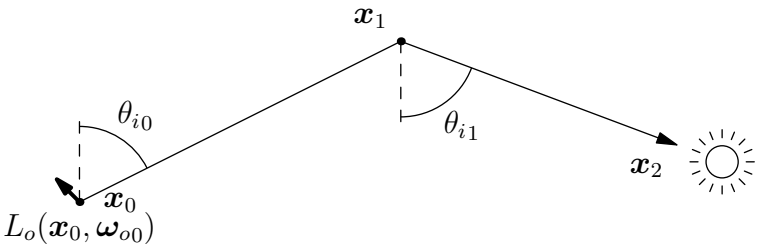


Figure 5.5: The simplest way to estimate the outgoing radiance $L_o(\boldsymbol{x}_0, \boldsymbol{\omega}_{o0})$ is eye tracing, which shoots a path from the surface into the scene and collects emissions at each intersection.

We can apply the observations made in section 5.3 to the path space formulation for radiance (equation (5.3)) to derive a path tracing based radiance estimator:

$$L_o(\boldsymbol{x}_0, \boldsymbol{\omega}_{o0}) \mathrel{\widehat{=}} \sum_{k=1}^{\infty} T_k L_e(\boldsymbol{x}_k, \boldsymbol{\omega}_{ok}), \quad T_n = \prod_{k=0}^{n-1} \frac{f(\boldsymbol{\omega}_{ik}, \boldsymbol{x}_k, \boldsymbol{\omega}_{ok}) \cos\theta_{ik}}{p(\boldsymbol{\omega}_{ik} \mid \boldsymbol{\omega}_{ok})} \tag{5.8}$$

This estimator alone however is ineffective at finding small light sources, so we combine it with Next Event Estimation through Multiple Importance Sampling (section 3.7), as we already did in

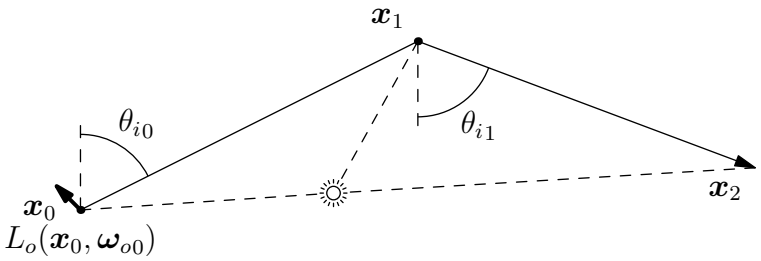


Figure 5.6: Direct light sampling also known as Next Event Estimation (NEE) performs significantly better than BSDF sampling on small direct lighting.

the reference pathtracer:

$$L_o(\boldsymbol{x}_0, \boldsymbol{\omega}_{o0}) \mathrel{\widehat{=}} \sum_{k=1}^{\infty} T_k \left(w_{\text{BSDF}} L_e(\boldsymbol{x}_k, \boldsymbol{\omega}_{ok}) + w_{\text{L}} \frac{f(\boldsymbol{x}_L \to \boldsymbol{x}_k, \boldsymbol{\omega}_{ok}) G(\boldsymbol{x}_k \leftrightarrow \boldsymbol{x}_L) L_e(\boldsymbol{x}_L \to \boldsymbol{x}_k)}{p(\boldsymbol{x}_L)} \right) \tag{5.9}$$

By combining the two techniques by weighting them with their respective MIS-weights $w_{\text{BSDF}}(\boldsymbol{\omega}_o)$ and $w_{\text{L}}(\boldsymbol{x}_L)$ (equation (3.27) and equation (3.28)), we get the best of both worlds. For brevity, the parametrization of the MIS-weights is dropped in the equation above. This combined with ReSTIR DI (Bitterli et al., 2020) and a LightBVH (Moreau et al., 2019) for efficient many light sampling is the radiance estimator used by Müller, Rousselle, et al. (2021). Nevertheless, I simply sample the light source from a precomputed CDF-table by inversion sampling, because my test scenes only contain a few light sources. To limit the path length, I use Russian Roulette Termination (equation (3.4)) again, and in praxis the path length also has to be limited to a maximum length, which introduces potential bias.

Query Prediction In practice, I only predict queries by shooting primary camera rays, because this is already a good approximation of the query sampling distribution $Q(q)$ for the path termination strategies given in section 4.4.1. A more accurate and efficient method however would be to randomly draw queries from the queries already generated in the previous inference stage.

5.5 Bidirectional Training (BT)

A major weakness of the aforementioned radiance estimator (section 5.4) is indirect lighting where the light paths undergo glossy reflections/refractions before getting diffused. This effect is known as a caustic. In the extreme case of an infinitesimal light source being ideally reflected/refracted and then diffused, path tracing and NEE even fail completely to capture this effect, because the probability to sample such a path by BSDF sampling is zero (figure 5.7a). However, we can do the exact opposite of NEE by tracing light paths and connecting them to query points sampled from the eye. This has the same problem as NEE for finding direct caustics (figure 5.7b), yet it



(a) NEE (b) Reverse NEE

Figure 5.7: Caustics are difficult to capture with path tracing, since only a small fraction of the path space actually contributes to the caustic (dotted lines). NEE does not help, as most connecting samples will have zero contribution (crossed line) when the material is glossy.

discovers indirect illumination by caustics and long light paths very efficiently.

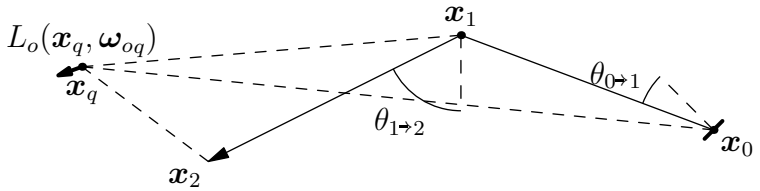


Figure 5.8: Flux is distributed along a BSDF-sampled path from a random point on a light source ($\boldsymbol{x}_0$). At the query point $\boldsymbol{x}_q$ radiance is collected by deterministically connecting to every vertex of the light path.

Specifically, I propose the following alternative radiance estimator based on this idea: First, sample a query point $q = (\boldsymbol{x}_q, \boldsymbol{\omega}_{oq})$ from the query sampling distribution $Q(q)$. Then, trace a light path carrying flux from a randomly selected light source via BSDF sampling and connect each vertex $\boldsymbol{x}_k$ to the query point $\boldsymbol{x}_q$.

Starting again with a single-sample Monte Carlo Estimator for the Path Space Integral for Radiance (equation (5.3)) and applying the same observations as in section 5.3, we get:

$$\begin{aligned} L_o(\boldsymbol{x}_q, \boldsymbol{\omega}_{oq}) \mathrel{\widehat{=}} & \frac{L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_q)}{p(\boldsymbol{x}_0)} G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_q) f(\boldsymbol{x}_0 \to \boldsymbol{x}_q, \boldsymbol{\omega}_{oq}) \\ & + \sum_{k=1}^{\infty} \Phi_k \cdot f(\boldsymbol{x}_{k-1} \to \boldsymbol{x}_k \to \boldsymbol{x}_q) G(\boldsymbol{x}_k \leftrightarrow \boldsymbol{x}_q) f(\boldsymbol{x}_k \to \boldsymbol{x}_q, \boldsymbol{\omega}_{oq}), \\ \Phi_n = & \frac{L_e(\boldsymbol{x}_0 \to \boldsymbol{x}_q) \cos\theta_{0\to1}}{p(\boldsymbol{x}_0) p(\boldsymbol{x}_0 \to \boldsymbol{x}_1)} \prod_{i=2}^{n} \frac{f(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1}) \cos\theta_{i\to i+1}}{p(\boldsymbol{x}_i \to \boldsymbol{x}_{i+1} \mid \boldsymbol{x}_{i-1} \to \boldsymbol{x}_i)}, \end{aligned} \tag{5.10}$$

where Φ_n is the light flux transported along the path. The sum over the path length is split into two parts because the first vertex $\boldsymbol{x}_0$ is sampled directly on the light source and requires special handling.

5.6 Light Training (LT)

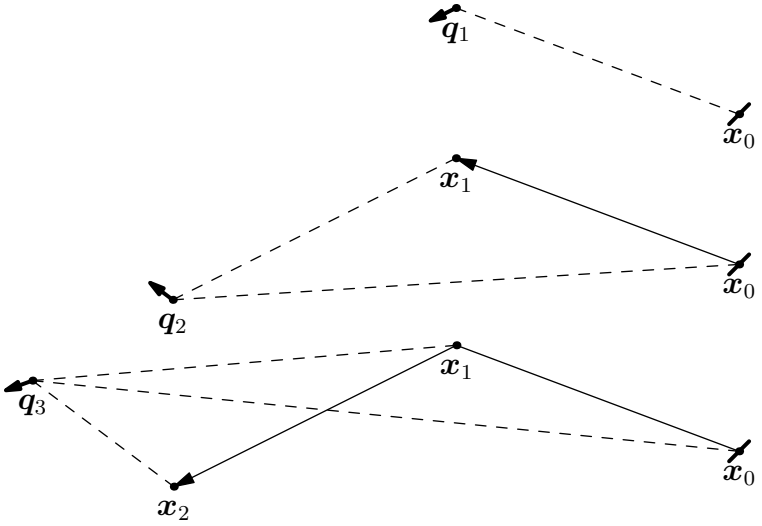
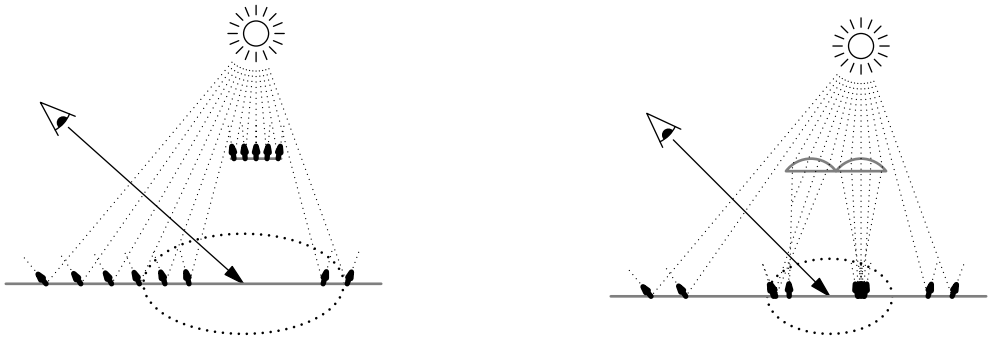


Figure 5.9: Visualization of light training for query prediction: Every new light vertex is first handled as a query (q_i) and collects radiance from all previous vertices according to equation (5.10).

Despite sampling light paths, the estimator from section 5.5 only finds indirect illumination by caustics efficiently, since given a specific query, the probability to connect to a glossy light path directly is still low (figure 5.7b). So, instead of predicting queries solely from the eye, we can also collect radiance at future vertices of the light path (see figure 5.9). At a light vertex, we are guaranteed to get strong contribution at least from the previous vertex, because we sampled it directly through BSDF sampling. It would also be possible to sample independent queries from the light vertices instead of reusing the next light vertex and this may in fact be an interesting algorithm to explore, though this is outside the scope of this thesis.

Balancing Although this estimator may look promising at first, it is important to note that it exhibits the problem already discussed in my motivation of bidirectional radiance caching (section 5.1): By sampling queries solely from the light, queries that are shadowed will never be predicted, which introduces bias into the cache (see figure 5.10). This can be clearly visible depending on the scene and will be further discussed in the results chapter (chapter 6).

To counteract this drawback, we can make a simple observation: If a query is not reachable from a light source, it does not receive energy, thus the radiance at such queries can be safely assumed to be zero. Because the Neural Radiance Cache performs averaging in local neighborhoods, we can carefully introduce samples with radiance zero according to the query sampling distribution $Q(q)$. By weighting the valid samples with the inverse of the ratio of valid samples, we can



(a) Shadows (b) Caustics

Figure 5.10: Sources of bias in light training: The solid arrows indicate cache queries. In both cases, the interpolation should return zero radiance, as the points are not reached by light. However, since the cache is filled by light tracing, all available nearby estimates are lit, resulting in a bright prediction.

ensure that the mean over a local neighborhood is still correct:

$$\mathbb{E}_q\left[\widehat{L}_o(\boldsymbol{x}_q, \boldsymbol{\omega}_{oq})\right] = \frac{1}{n}\sum_{q\in Q}(1-p_0)\cdot\frac{L_o(\boldsymbol{x}_q, \boldsymbol{\omega}_{oq})}{(1-p_0)} + p_0\cdot 0 = \frac{1}{n}\sum_{q\in Q} L_o(\boldsymbol{x}_q, \boldsymbol{\omega}_{oq}) \tag{5.11}$$

Essentially this algorithm models a sort of output decay on the NRC, dragging its prediction towards zero. This can work, but it degrades the quality of the cache and makes training unstable, since it introduces additional noise with the zero radiance samples. Furthermore, it is a waste of bandwidth to spend resources on learning zero radiance, as it does not provide any information gain.

5.7 Sparse Progressive Photon Collection (SPPC)

The final radiance estimator I propose is an adaptation of Progressive Photon Mapping by Hachisuka, Ogaki, et al. (2008) and Jensen (1996). We again shoot flux into the scene by light tracing. However, instead of collecting the flux by connecting to an infinitesimal query point, we collect the incoming flux over all sampled light paths in a *query region* by kernel density estimation (see figure 5.11).

The radiance at a query point $\boldsymbol{x}$ in direction $\boldsymbol{\omega}_o$ can be estimated by:

$$L_o(\boldsymbol{x}, \boldsymbol{\omega}_o) = \int_{\Omega_i} f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) L_i(\boldsymbol{\omega}_i, \boldsymbol{x}) \cos\theta_i \, d\boldsymbol{\omega}_i \tag{5.12a}$$

$$= \int_{\Omega_i} f(\boldsymbol{\omega}_i, \boldsymbol{x}, \boldsymbol{\omega}_o) \frac{d^2\Phi_i(\boldsymbol{\omega}_i, \boldsymbol{x})}{dA \cancel{\cos\theta_i \, d\boldsymbol{\omega}_i}} \cancel{\cos\theta_i \, d\boldsymbol{\omega}_i} \tag{5.12b}$$

$$\widehat{=} \frac{1}{n} \sum_{k=1}^{n} f(\hat{\boldsymbol{\omega}}_{ik}, \boldsymbol{x}, \boldsymbol{\omega}_o) \frac{\Delta\Phi_i(\hat{\boldsymbol{\omega}}_{ik}, \hat{\boldsymbol{x}}_k)}{\Delta A} w_{\boldsymbol{x}}(\hat{\boldsymbol{x}}_k) \tag{5.12c}$$

$$\approx \frac{1}{n\pi r^2} \sum_{k=1}^{n} f(\hat{\boldsymbol{\omega}}_{ik}, \boldsymbol{x}, \boldsymbol{\omega}_o) \Delta\Phi_i(\hat{\boldsymbol{\omega}}_{ik}, \hat{\boldsymbol{x}}_k) w_{\boldsymbol{x},r}(\hat{\boldsymbol{x}}_k), \tag{5.12d}$$

where $w_{\boldsymbol{x},r}(\hat{\boldsymbol{x}}_k)$ is the kernel weight for the photon position $\hat{\boldsymbol{x}}_k$. We start with the rendering equation (equation (5.12a)) and replace the incoming radiance L_i by its definition as incoming flux per area and solid angle (equation (5.12b)). We then replace the integral by a Monte Carlo estimator over n sampled flux packages *(photons)* (equation (5.12c)). Finally, we assume that the kernel function $w_{\boldsymbol{x},r}(\hat{\boldsymbol{x}}_k)$ is limited to a sphere of radius r. Assuming the surface is locally flat, this allows us to approximate the surface area ΔA over which we are integrating by a disk of area $A = \pi r^2$ (equation (5.12d)). The simplest such kernel function is the box function, with $w_{\boldsymbol{x},r}(\hat{\boldsymbol{x}}_k) = 1$ inside the sphere and 0 outside.

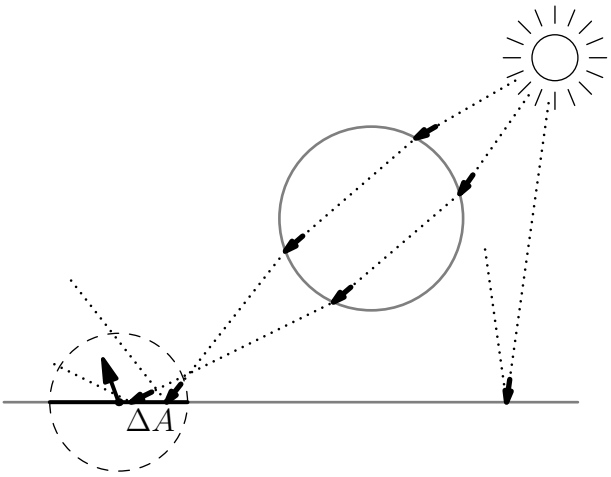


Figure 5.11: Visualization of photon mapping. Photons (packages of flux, here small black arrows) are distributed in the scene and collected over query regions (dashed circle). In this example, three light paths were sampled ($n = 3$). Note, that in practice photons and queries are not recorded at glossy surfaces, since they probably can not be connected with high contribution. This removes glossy features from the radiance estimate, however the path termination strategies from section 4.4.1 avoid querying glossy surfaces in the first place.

Query Lifetimes Because we couple Photon Mapping with radiance caching, we can make some important modifications to the original algorithm: Firstly, we do not need to query the

Table 5.1 The structs used to represent photons and queries. Unit vector compression strategies (Cigolle et al., 2014) could be applied to reduce the memory footprint. To keep the code simple, this is not done in the accompanying implementation.

(a) Photon

Field	Description
$\boldsymbol{x} \in \mathbb{R}^3$	Position
$\boldsymbol{\omega}_i \in \mathbb{S}^2$	Incident direction
$\boldsymbol{\Phi}_i \in \mathbb{R}^3$	Flux
$\boldsymbol{n} \in \mathbb{S}^2$	*Surface normal for normal rejection (optional)*

(b) Query

Field	Description
$\boldsymbol{x} \in \mathbb{R}^3$	Position
$\boldsymbol{\omega}_o \in \mathbb{S}^2$	Outgoing direction
$\boldsymbol{\Phi}_o \in \mathbb{R}^3$	Accumulated outgoing flux
$r \in \mathbb{R}$	Radius
$\boldsymbol{n} \in \mathbb{S}^2$	Surface normal at $\boldsymbol{x}$
$f : \mathbb{S}^4 \to \mathbb{R}^3$	BSDF at $\boldsymbol{x}$
$n_c \in \mathbb{N}$	Number of total light samples at creation
$N \in \mathbb{R}$	Amount of photons collected in all passes
$M \in \mathbb{N}$	Number of photons collected in current pass

photon map at every pixel, since the radiance cache interpolates between sparse samples. So, we can reduce the amount of queries to a fixed number independent of render resolution. Secondly, we can store the *queries* instead of the *photons* in a spatial data structure. This enables the queries to persist across multiple frames and collect incoming flux over multiple light passes. The lifetimes of the queries can be made completely independent of each other by keeping track of the total amount of light samples already drawn at creation time of a query. The difference between the current count of light samples and the count at creation gives the number of total light samples n that are used in the collection of flux for that query (equation (5.12)). Flux is simply accumulated to the query whenever a photon hits the query region. To maintain a constant number of queries I use a ring buffer with an atomic index counter and a constant query replacement fraction κ. The lifetime of a query is thus approximately $1/\kappa$ frames, yet it can also be longer if query prediction fails to produce a valid result, for example if a camera ray does not hit any geometry.

Radius Reduction Choosing the radius r is a trade-off between variance and bias. Larger radii collect flux over a larger area, increasing the number of photons used in the estimator and thus reducing variance. However, this also introduces bias, as these additional photons may not be an accurate representation of the incoming flux at the query point. Shrinking the radius r towards zero eliminates this bias in the limit, but the number of available estimates also decreases, increasing the variance. More precisely, Knaus and Zwicker (2011) showed that $\text{Bias}[\langle L_{o\text{PM}} \rangle_N] \sim r^2 \sim 1/\text{Var}[\langle L_{o\text{PM}} \rangle_N]$.

Leveraging this observation, Hachisuka, Ogaki, et al. (2008) developed Progressive Photon

Mapping (PPM), originally as a technique to make classic Photon Mapping consistent and to overcome the memory constraint of storing a single large photon map. Their idea was to perform multiple light passes, shrinking the radius after each iteration, yet never too much, as there has to be a gain in the total amount of photons, otherwise the estimator diverges. Therefore, they introduce a factor α to control the fraction of photons to keep after each light pass (equation (5.13a)). By fixing the photon density inside the area of the query (equation (5.13b)), they derive following update rules:

$$\text{Let } N' = N + \alpha M \tag{5.13a}$$

$$\frac{N+M}{A} = \frac{N+M}{\pi r^2} \overset{!}{=} \frac{N'}{\pi (r^2)'} = \frac{N'}{A'} \implies (r^2)' = r^2 \frac{N'}{N+M} \tag{5.13b}$$

$$\Phi_o' = \Phi_o \frac{A'}{A} = \Phi_o \frac{\pi (r^2)'}{\pi r^2} = \Phi_o \frac{N'}{N+M}, \tag{5.13c}$$

where N is the total amount of collected photons, M is the number of new photons found during the light pass, and r and A are the respective radius and area of the query. The total outgoing flux has also to be scaled down to the reduced area A' (equation (5.13c)).

Hardware Acceleration The original k-d-tree method of storing photons by Jensen (1996) is not suitable for GPUs. Thus, this implementation instead follows the idea of Evangelou et al. (2021), who proposed to leverage hardware accelerated Bounding Volume Hierarchies (BVHs) for fast Fixed-Radius-Near-Neighbor queries. Their approach is to store the queries as balls of radius equal to their search radius r_i and to represent the data points by shooting infinitesimal rays. Finding an intersection is then equivalent to finding a data point within r_i around the query.

Algorithm Overview Applied to Photon Mapping, this means we first store the Axis-Aligned Bounding Boxes (AABBs) of the queries in an array and use OptiX to construct a BVH. Then, we sample the light paths and at each non-specular scene intersection (section 4.4.1), we shoot infinitesimal rays into the query-BVH. In the intersection program, we simply compare the squared distance to the ray origin with the squared radius of the query ball. If the distance is smaller, we atomically accumulate the BSDF-weighted incoming flux of the photon to the flux of the query Φ_o. After the light pass we apply radius reduction and use the radiance estimates to update the radiance cache.

Notes on Performance If the queries are dense or the radius is large, many queries will overlap, causing the atomic accumulation to be serialized and stalling the GPU threads. This is the reason why Kern et al. (2023) store the *photons* instead of the queries in the BVH, which forces them however to use a global radius reduction scheme (Knaus and Zwicker, 2011). Yet, because we allow the queries to have independent lifetimes, a global radius is not applicable in our case, rendering their approach incompatible. Fortunately though, we partially avoid serialization since our queries are only distributed sparsely. An interesting idea to resolve this problem would

be to check for overlap during query prediction and reject queries that overlap too much with existing queries, or to use an iterative screen-space hole-filling approach like Stachowiak (2018). This would also automatically sample caustics more densely, as caustics receive more light samples, making the radius shrink faster, thus reducing overlap and creating space for new queries. Additionally, Kern et al. (2023) stochastically reject 70% of all non-caustic photons, which could also benefit performance in our case. Furthermore, to improve quality at corners, they reject photons with strongly deviating normals.

6 Results

6.1 Methodology

For the purpose of reproducible evaluation, a JSON-configurable CLI tool was used to generate all results presented in this thesis and the exact parametrization is distributed along the source code on the accompanying repository (Stamm, 2025).

Network Architecture For the training and inference, the fully fused network architecture introduced by Müller, Rousselle, et al. (2021) and described in section 4.1 is used. The implementation uses the *tiny-cuda-nn* library (Müller, 2021) that was distributed alongside the original NRC paper and updated to also provide an implementation of the Multiresolution Hash Encoding (MHE) presented by Müller, Evans, et al. (2022). The network architectures are kept identical to those in the original papers. Specifically, the MHE network uses 3 hidden layers with 64 neurons, while the Triangle Wave Encoding variant uses 5 hidden layers instead to compensate for the lower quality of the encoding.

Performance Measurement Cuda Events are used to evaluate performance. Tone mapping and screen blitting are not included into the measurements, so only the pure rendering time is reported. If not otherwise stated, all measurements are taken on an NVIDIA RTX 3060 Ti GPU and averaged over at least 10 seconds.

Error Metric The reference images are generally generated by Path Tracing (PT), except for the Caustics scene, which is rendered by Stochastic Progressive Photon Mapping (SPPM) (Hachisuka and Jensen, 2009) since PT does not converge in reasonable time. To compare the quality of different techniques, the Mean Squared Error (MSE) in linear RGB space is reported for numerical error and the HDR variant of the ꟻLIP metric (Andersson et al., 2021) for humanly perceived error. Furthermore, bias and variance are measured separately by computing the first and second moment over 1,000 independent 1spp renders. Between the bias and variance samples the configuration and cache are kept constant, but for the inference pass different samples are drawn from a Quasi-Random Low Discrepancy Sequence.

Scenes Tests are performed on a set of scenes that are also included in the source code repository (Stamm, 2025):

- The DIFFUSE scene is an adaptation of the Cornell Box and highlights diffuse indirect illumination.
- The THINKER scene incorporates glossy and transmissive materials with complex specular highlights. It contains decimated and remeshed version of the Thinker statue by Auguste Rodin scanned by Scan the World (2014) and the Stanford Bunny scanned by the Stanford University Computer Graphics Laboratory (1994).
- The CHESS scene is a complex textured scene composed of public domain assets from Poly Haven (2025).
- The AJAR scene is mostly lit by long indirect light paths, which are difficult to find with eye tracing.
- The CAUSTICS scene contains complex sharp caustic patterns that are challenging for traditional path tracing.

6.2 Parametrizing the NRC

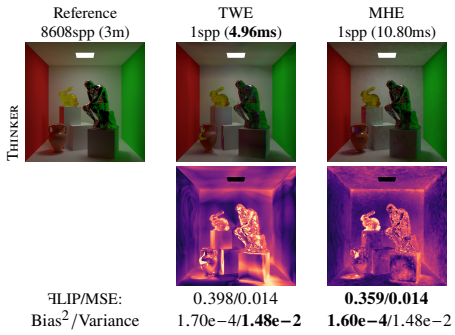


Figure 6.1: Comparison of the different input encodings.

Input Encodings The input encoding has a significant impact on both quality and performance of the cache (see figure 6.1). The Multiresolution Hash Encoding (MHE) by Müller, Evans, et al. (2022) comes at a great performance hit compared to the original Triangle Wave Encoding (TWE) by Müller, Rousselle, et al. (2021) (more than 2× in the test), but distinctly enhances the representation of high-frequency features. Since we aim to capture caustics, the MHE is the natural choice used in the following evaluations.

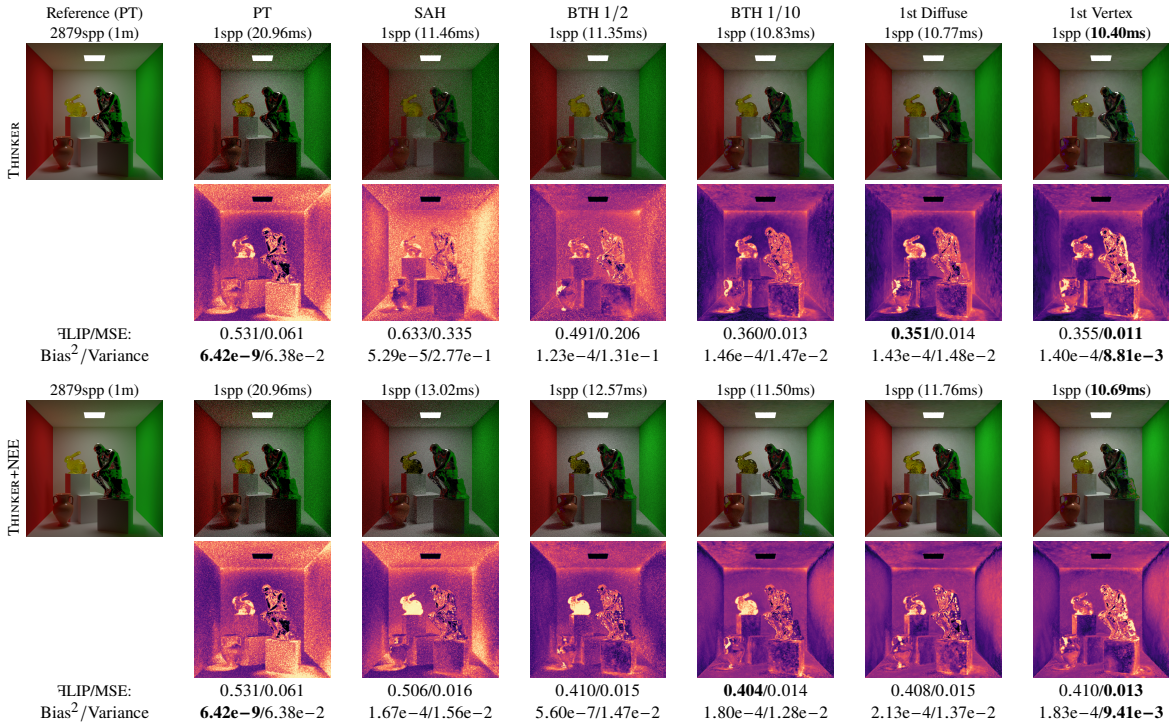


Figure 6.2: Comparison of the Path Termination strategies from section 4.4.1.

Path Termination Strategies Choosing a path termination strategy is a trade-off between bias and variance (see figure 6.2). The SAH termination strategy has the highest variance and also the highest performance cost, because it terminates paths the least aggressively. On the other end of the spectrum, terminating on the first vertex directly has the lowest variance and is also the fastest but has some issues with specular highlights. All the other strategies lie somewhere in between, with the parametrized BTH strategy bridging the gap. As the SPPC strategy does not learn glossy highlights at all and since the quality of the NRC seems good enough for early termination, the following tests use termination at the first diffuse vertex. Combining the inference with NEE did only benefit the less aggressive termination strategies, so the rest of the test only use BSDF sampling for inference.

Training Like recommended by Müller, Evans, et al. (2022), training is performed using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-2}, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-15}$ and an L2 regularization factor of 10^{-6}. Furthermore, they use 4 training steps with a batch size of $2^{14} = 16,384$ each. Tests show, that the usage of multiple training steps per frame leads to faster cache convergence (see figure 6.3) while larger training sets evidently result in lower error. However, multiple training steps also slow down performance. The following tests use single training steps of size $2^{17} = 131,072$ to minimize error.

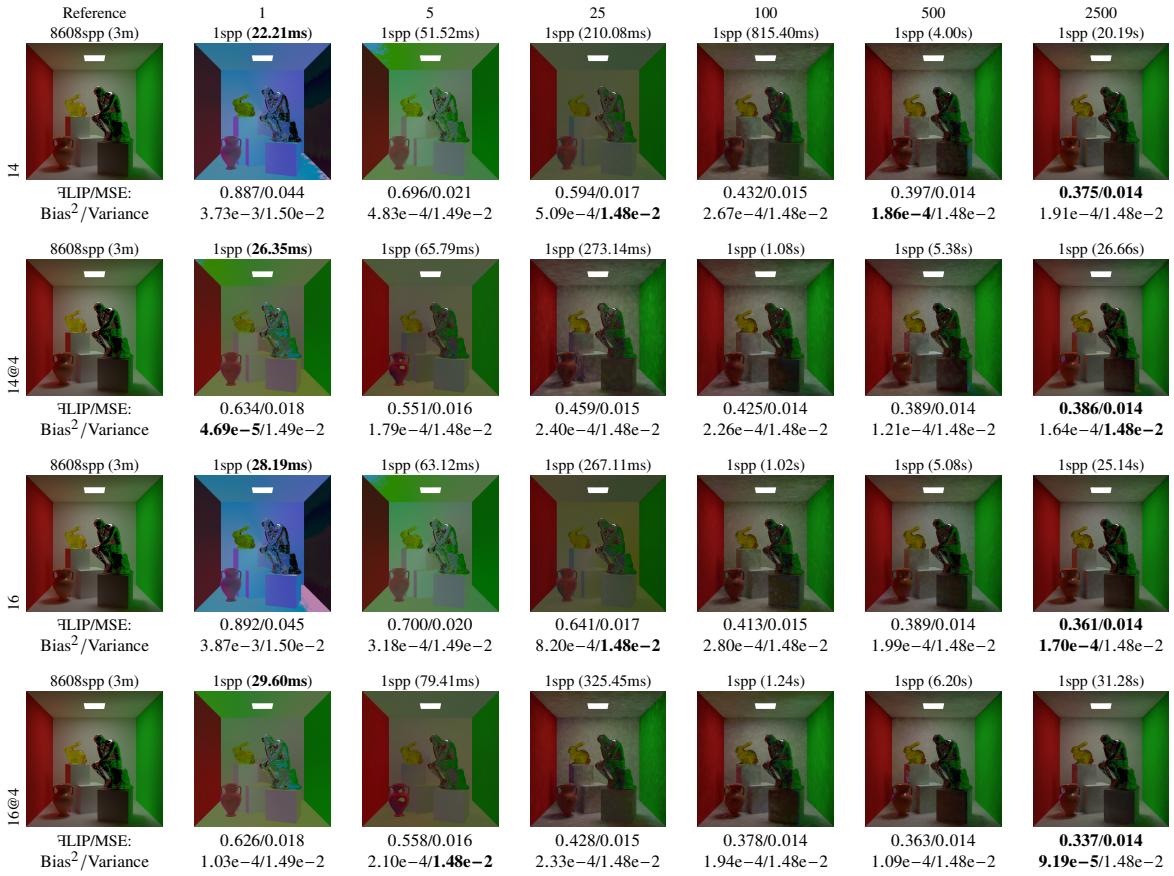


Figure 6.3: Comparison of cache convergence for different training set sizes (14: $2^{14} = 16,384$, 16: $2^{16} = 65,536$) and training steps per frame (@4 indicates that the training set is split into 4 batches, otherwise it is processed as a single batch). The columns indicate how many frames were rendered to train the cache (1, 5, 25, 100, 500, 2500). Above the images, the numbers in parentheses denote the total time spent on training and inference.

6.3 Optimizations

Kernel Fusion Recently, the *tiny-cuda-nn* library of Müller (2021) was extended by a feature that allows combining the encoding, MLP, loss function, training and inference steps together into fused monolithic kernels by on-the-fly JIT compilation. Using this approach, Müller (2021) reports a potential speedup of 1.5× up to 2.5× for training and inference, especially for modern GPU architectures. Measurements on an NVIDIA RTX 3060 Ti, however, show a performance *decrease* in the inference step and similar performance for training (see figure 6.5). This could potentially be caused by an older GPU architecture, the following tests are thus performed without JIT-fusion.

Photon Mapping To optimize Photon Mapping, several optimizations were tested on full SPPM (see figure 6.6). The biggest performance improvement was achieved by stochastically

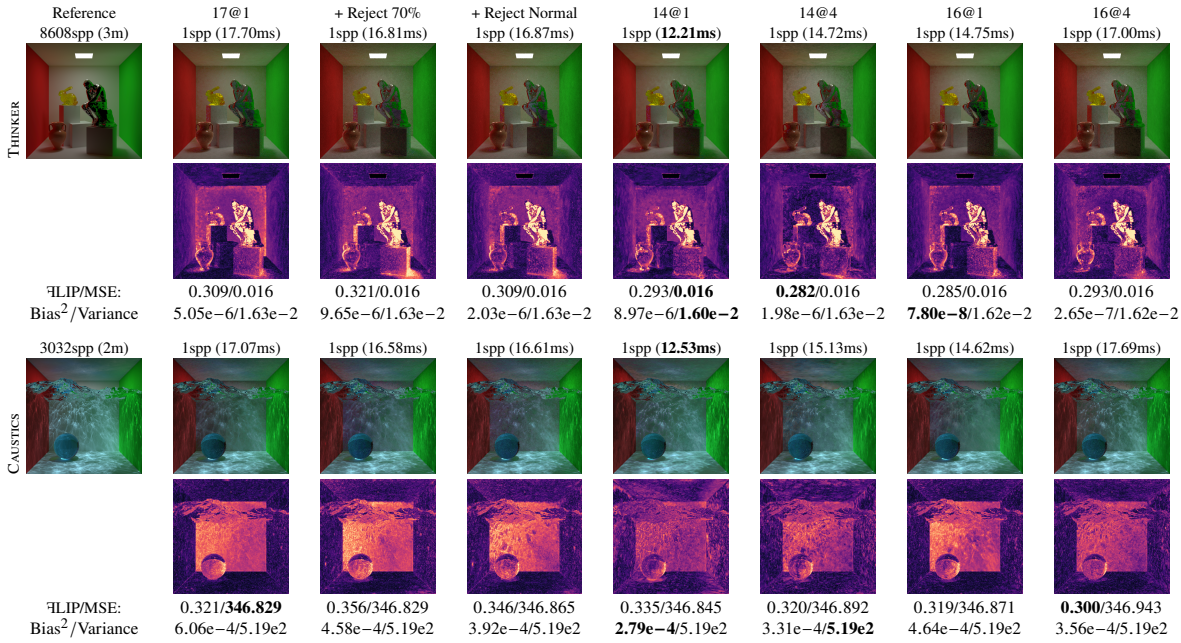


Figure 6.4: Error and bias for SPPC with different configurations. 17@1 indicates a cache size of $2^{17} = 131,072$ and 1 training step per frame, 14@4 indicates a cache size of $2^{14} = 16,384$ and 4 training steps per frame, etc.

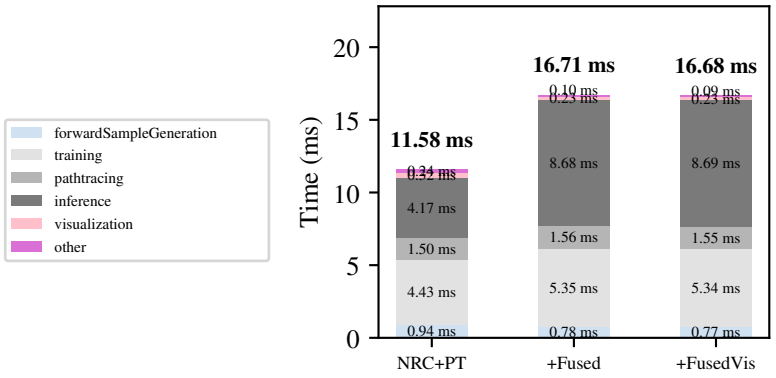
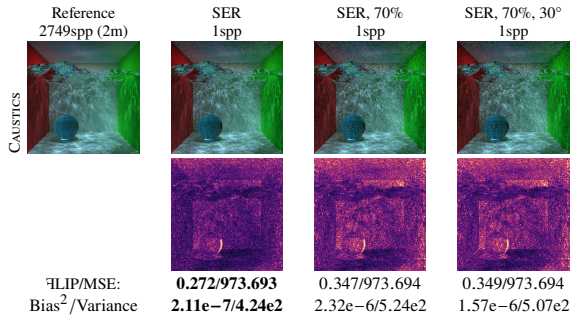


Figure 6.5: Applying JIT kernel fusion. *+Fused* indicates fusion of the encoding with the training and inference kernels, *+FusedVis* also fuses the visualization step, where the output of the network is multiplied with the reflectance and accumulated to the render buffer. Unfortunately, enabling JIT-fusion *decreases* performance, probably because of the older GPU architecture.

rejecting a significant portion of the non-caustic photons, as proposed by Kern et al. (2023). This however decreases quality in non-caustic areas, and the performance gain does not carry over to SPPC (see figure 6.4) where the overlap of queries is limited, so we do not use rejection for evaluation. In the following tests, 100,000 photons are traced per frame, with an initial query radius of 0.001, a radius reduction factor of $\alpha = 0.7$ and a query replacement factor of $\kappa = 0.001$,



Technique	Frametime	Change
Baseline	**159.50ms**	
IS only	178.19ms	+11.7%
SER	158.54ms	−0.6%
SER, 70%	124.46ms	−22%
SER, 70%, 30°	123.58ms	−22.5%

Figure 6.6: Optimizations to Photon Mapping. *IS only* directly accumulates in the Intersection Shader (IS) and skips the Any Hit Shader (AH). *SER* uses Shader Execution Reordering between the Intersection and Any Hit shader to improve coherence. *SER, 70%* additionally rejects 70% of the non-caustic photons in the Any Hit shader (Kern et al., 2023). *SER, 70%, 30°* rejects photon hits whose surface normals deviate more than 30° from the surface normals at the query point (Kern et al., 2023). The most notable performance improvement comes from rejecting non-caustic photons, however, this notably increases error in non-caustic areas. Rejecting photons based on normal deviation has a small positive performance impact and decreases bias. Timings are averaged over 16 runs.

resulting in an approximate query lifetime of $1,000$ frames.

6.4 Evaluation

Quality Figure 6.11 compares the different radiance estimation techniques from chapter 5 with respect to quality. Applying self learning (SL) to the original NRC method degraded quality in almost all scenes except for the Caustics scene. This could be caused by a faulty implementation, since Müller, Rousselle, et al. (2021) report a *decrease* in bias when applying self learning. Bidirectional training (BT) was consistently either faster or comparably fast to the original PT method and even decreased perceived error in all scenes except the Chess scene. Light training (LT) on the other hand, did not produce usable results. Finally, the SPPC estimator produced the lowest perceived error in all scenes, except in the Thinker scene, where the glossy specular highlights exhibit a bright bias. This should however normally be covered by the path termination strategy continuing the querying paths, so this may likely be caused by a bug in the implementation of the inference phase. The improved quality comes with an overhead of around $3-5$ms per frame but still stays in the budget for interactive rendering (16ms).

Figure 6.7 takes a closer look at light training. Different balancing strategies were tested to counteract the bias, *Surface Balancing* generates zero samples uniformly on the surfaces in the scene, while *Camera Balancing* generates zero samples by shooting primary camera rays. Both times, the samples are mixed in at a ratio of 1:1 with the light training samples, so the light training samples are weighted twice as high. Unfortunately, neither of these strategies achieve a

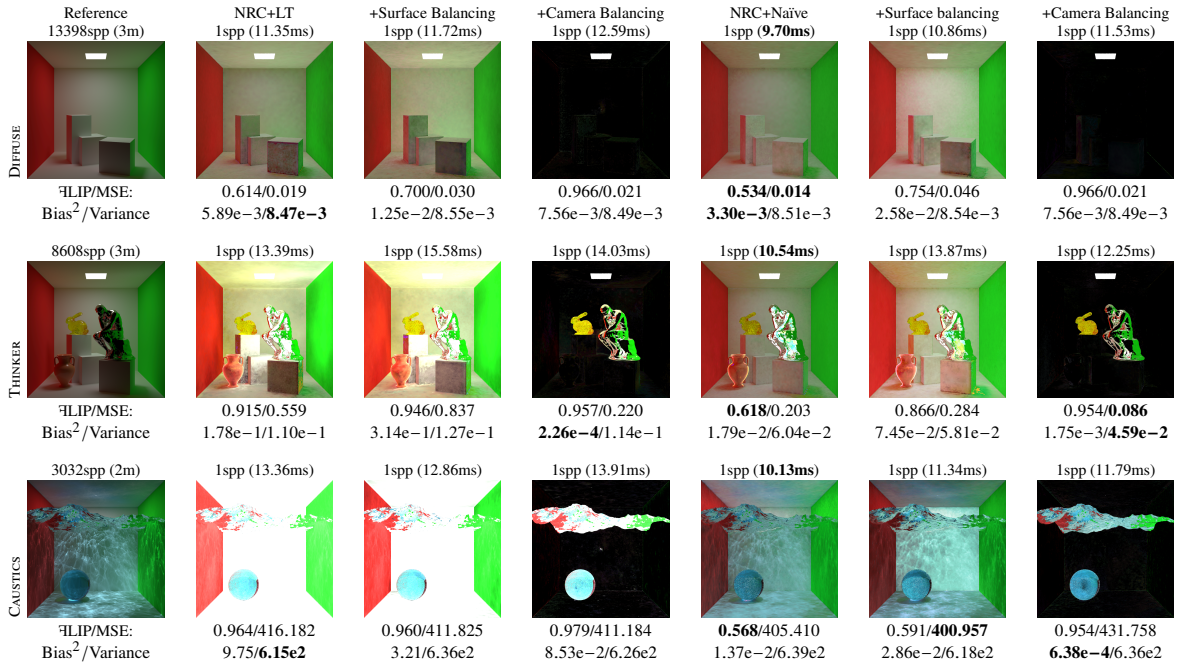


Figure 6.7: Comparing light training and balancing strategies specifically.

stable mixing ratio, with surface balancing dragging the cache too little towards zero and camera balancing dragging it too much. Light training is also compared to simply storing the outgoing flux at the light path vertices (*Naïve*). Surprisingly, this produces better results than light training, even with the technique not being a valid radiance estimator. Yet, it is still too far from the reference to be usable.

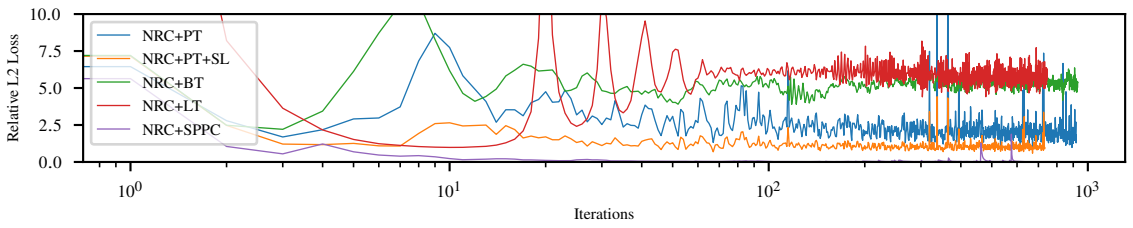


Figure 6.8: Comparison of training convergence. Because the training is online, there is no ground truth validation step. The loss thus mainly indicates the variance of the training set. BT and LT have the highest training errors because both utilize light sampling, SPPC has the lowest training error because it keeps estimates in a fixed ring buffer for accumulation. Measured on the Thinker scene.

Performance Figure 6.9 compares the performance of the different radiance estimation techniques. All the NRC-based techniques are faster than pure path tracing (PT) since they terminate paths early and only evaluate long paths sparsely. The bidirectional training (BT) and the original method come in at around the same performance cost. Self-learning (SL) adds a

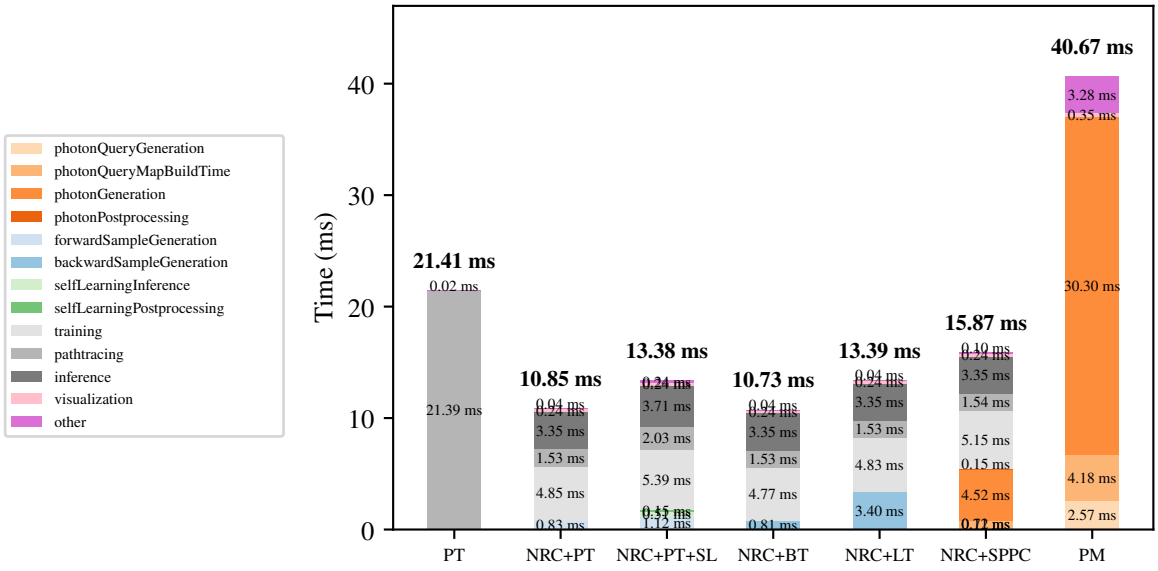


Figure 6.9: Performance breakdown of the different rendering techniques. Measured in 720p-equivalent 960^2px resolution on the THINKER scene on an NVIDIA RTX 3060 Ti and averaged over 10s. The *other* category represents memory and timing operations.

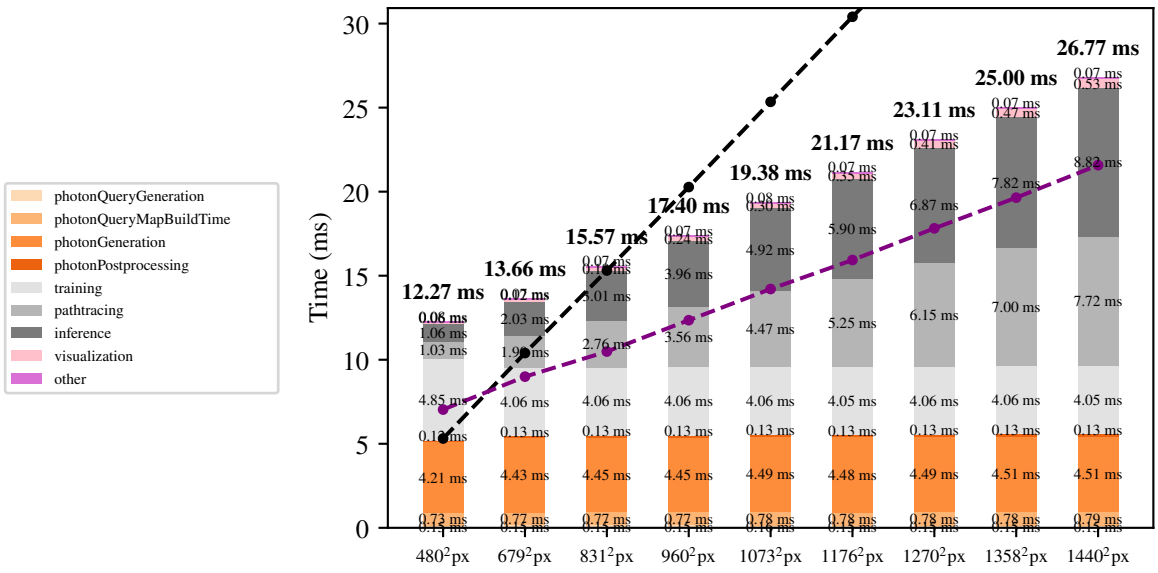


Figure 6.10: Performance vs resolution of SPPC. For reference, the performance of PT (black line) and NRC+PT (purple line) is also shown. The resolution sequence is linear in total pixel count, going up to 1080p-equivalent 1440^2px. Measurements are averaged over 10s. Note, that PT has a near zero constant overhead, but a significantly higher per-pixel cost, whereas both Radiance Caching techniques have a much lower per-pixel cost but higher overhead.

small overhead of around 0.5ms per frame caused by the additional inference step. SPPC is the

slowest NRC-based technique, but still considerably faster than the unoptimized implementation of full SPPM. Moreover, its performance can be controlled by changing the number of photons traced per frame, applying stochastic rejection and tweaking the initial query radius.

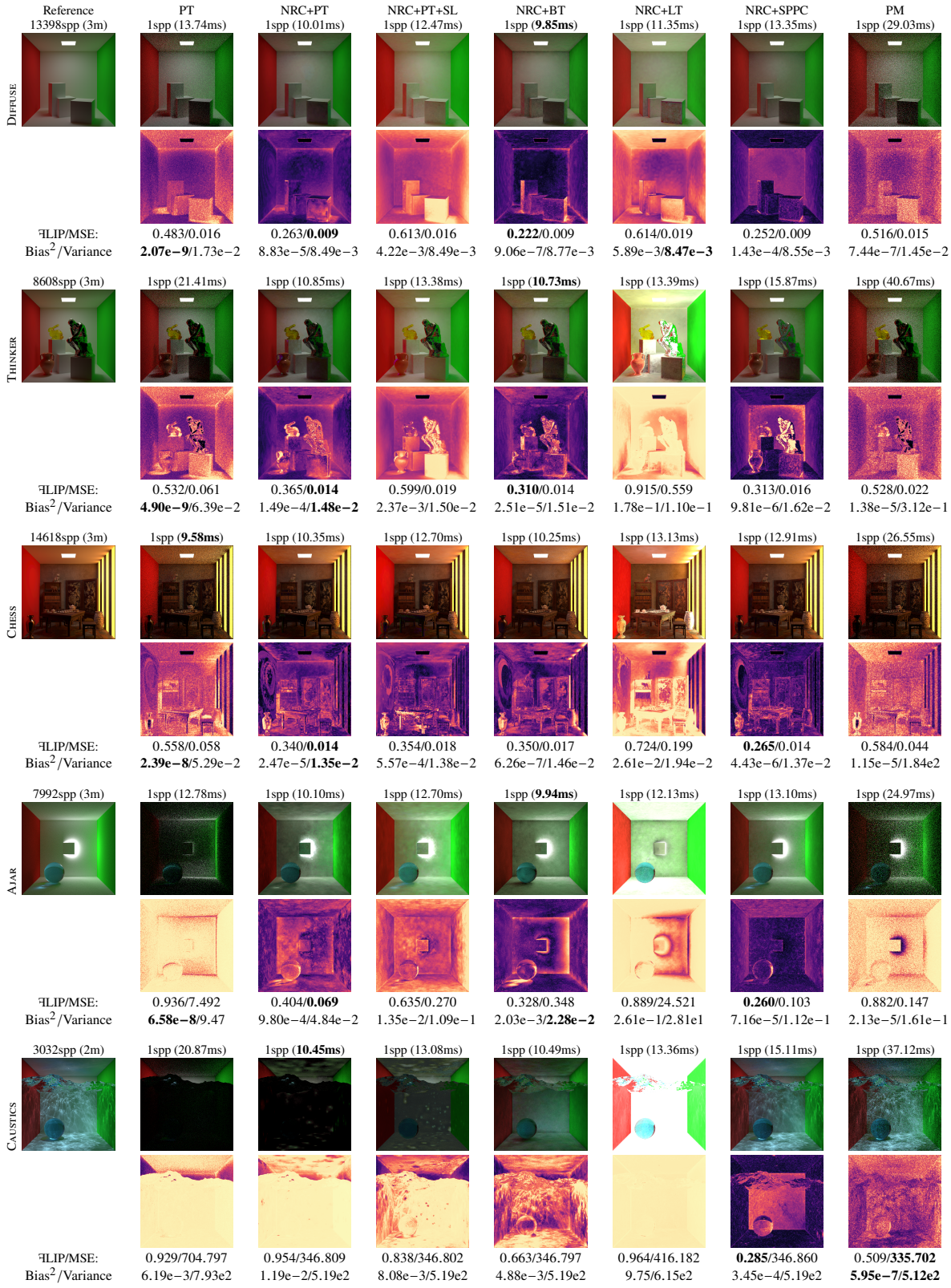


Figure 6.11: Comparison of the different radiance estimators from chapter 5. To isolate training quality, inference is terminated after the first diffuse vertex and is not combined with NEE.

7 Conclusion

7.1 Summary

In this thesis, I presented and derived three new radiance estimation techniques to train the Neural Radiance Cache. Evaluation showed, that two of these three proposed radiance estimators improve upon the original paper, especially in the area of caustics and indirect illumination.

The bidirectional estimator performs very comparable to NRC, while improving the quality of indirect illumination. It struggles, however, with direct lighting and caustics, because the probability of finding a matching pair of a query and a directly contributing light path is low. Nevertheless, it is a competitive alternative to the PT-based training from the original paper (Müller, Rousselle, et al., 2021).

The light training estimator has potential to capture sharp details, but suffers from bias which is caused by not sampling shadowed areas. Unfortunately, this makes it largely unusable in practice.

The estimator based on Progressive Photon Mapping is particularly robust and produces high quality caustics and indirect illumination, while coming only at a mild performance overhead compared to training by PT. This one is particularly promising for practical use cases, since it can produce fairly accurate and noise free renderings of caustics in real-time, which can neither be achieved by pure path tracing nor by the original NRC. It comes, however, at the cost of sharpness and time lag, thus it would be interesting to evaluate its performance in dynamic scenes. Another limitation with this and also the other estimators is, that it is better suited for indoor scenes, as in large-scale or outdoor scenes unguided light sampling would cover large areas not directly visible to the camera. In these cases, light culling or guiding would be necessary.

7.2 Further Optimization

Query Prediction Currently, query points are predicted mostly by shooting primary camera rays. A better approach would be to stochastically store queries during the inference phase and use these in the following training pass to estimate radiance. This would not only be a more accurate

approximation of the future querying, it would also likely be lighter on performance because it prevents redundant intersection tests. The only downside could be a one frame latency in query prediction, but because the NRC inference is already filtered by an Exponential-Moving-Average this is likely negligible.

Another interesting approach could be to target radiance estimations at prioritized areas in the scene, e.g. in areas that received illumination changes by object or light movement or areas that are focused by the user's attention (for example by gaze detection).

SPPC As already discussed in section 5.7, synchronization of the atomic accumulation is likely the biggest factor that holds back the performance of hardware accelerated photon mapping. Thus, it could be interesting to either explore stochastic evaluation techniques like Kern et al. (2023) or to reduce overlap during query prediction, similar to what e.g. Stachowiak (2018) did for surfel placement. Warp Aggregation is another common technique to resolve this in classical GPU programming by first aggregating subsets and then reducing, however OptiX does not expose warps and the threads can be unordered, making this not straightforward.

7.3 Future Work

By generalizing Neural Radiance Caching, this thesis opens a wide array of possible future improvements.

Radiance Estimators Particularly interesting is the integration of different radiance estimators. Although the SPPC estimator already produces high quality caustics, the density estimation step of Photon Mapping reduces sharpness of high-frequency features and introduces bias at edges. It could be interesting to use e.g. Vertex Connection and Merging (Georgiev et al., 2012), Metropolis Light Sampling (Veach and Guibas, 1997), full MIS-weighted Bidirectional Path Tracing (Lafortune and Willems, 1993; Veach, 1997) or Manifold Exploration techniques (Jakob and S. Marschner, 2012) to generate training data. The proposed framework for Radiance Caching does not impose any restrictions on *mixing* different radiance estimators, so independently choosing the best suited radiance estimation strategy for different areas of the scene could also prove interesting.

Radiance Caches The Neural Radiance Cache is fairly expensive to train and infer. The MHE encoding significantly improves quality, but the benefit largely comes from the underlying hash grid. Thus, manually crafted grid based approaches could also be promising and lighter on performance.

It would be interesting to combine the SPPC estimator with other caches, e.g. based on spherical harmonics (Křivánek, Gautron, Pattanaik, et al., 2005).

It may also be possible to derive a better radiance cache by storing the incoming radiance and BSDFs in a data structure similar to Virtual Point Lights in Instant Radiosity techniques (Keller, 1997) and connecting to them stochastically through e.g. ReSTIR DI (Bitterli et al., 2020). For Virtual Point Lights, this has already been explored by Bruin (2025).

Bibliography

Áfra, A. T. (2019). "Open Image Denoise". In: Intel Corporation. URL: https://www.highperformancegraphics.org/wp-content/uploads/2019/hot3d/open_image_denoise.pdf (visited on 08/25/2025).

Andersson, P., J. Nilsson, P. Shirley, and T. Akenine-Möller (May 2021). "Visualizing Errors in Rendered High Dynamic Range Images". In: *Eurographics Short Papers*. URL: https://research.nvidia.com/publication/2021-05_visualizing-errors-rendered-high-dynamic-range-images.

Apers, D. (2024). "Shipping Dynamic Global Illumination in Frostbite". In: *ACM SIGGRAPH 2024 Courses: Advances in Real-Time Rendering in Games*. SIGGRAPH 2024. URL: https://advances.realtimerendering.com/s2024/content/EA-GIBS2/Apers_Advances-s2024_Shipping-Dynamic-GI.pdf (visited on 05/04/2025).

Arvo, J. et al. (1986). "Backward Ray Tracing". In: *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*, pp. 259–263.

Bekaert, P., P. Slusallek, R. Cools, V. Havran, and H.-P. Seidel (2003). "A Custom Designed Density Estimation Method for Light Transport". In: URL: https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_1819214 (visited on 08/03/2025).

Binder, N., S. Fricke, and A. Keller (Feb. 3, 2021). *Massively Parallel Path Space Filtering*. arXiv: 1902.05942 [cs]. URL: http://arxiv.org/abs/1902.05942 (visited on 08/25/2025). Pre-published.

Bitterli, B., C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz (Aug. 31, 2020). "Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting". In: *ACM Transactions on Graphics* 39.4. URL: https://dl.acm.org/doi/10.1145/3386569.3392481 (visited on 06/25/2025).

Blender Foundation (2025). *Sampling Patterns - Blender Developer Documentation*. URL: https://developer.blender.org/docs/features/cycles/sampling_patterns/ (visited on 06/15/2025).

Boissé, G., S. Meunier, H. de Dinechin, P. Bartels, A. Veselov, K. Eto, and T. Harada (Oct. 30, 2023). *GI-1.0: A Fast and Scalable Two-level Radiance Caching Scheme for Real-time Global Illumination*. arXiv: 2310.19855 [cs]. URL: http://arxiv.org/abs/2310.19855 (visited on 08/02/2025). Pre-published.

Bruin, S. (2025). "Global Illumination Using ReSTIR DI and Photon-Mapped Virtual Point Lights". PhD thesis. Delft University of Technology. URL: https://repository.tudelft.nl/record/uuid:ddb7ccdd-289b-48a4-bffa-541840f9cfbf (visited on 08/21/2025).

Burley, B. (2020). "Practical Hash-based Owen Scrambling". In: *Journal of Computer Graphics Techniques* 10.4, p. 29. URL: https://jcgt.org/published/0009/04/01/paper.pdf.

Burley, B. and Walt Disney Animation Studios (2012). "Physically-Based Shading at Disney". In: *Acm Siggraph*. SIGGRAPH. Vol. 2012. 2012. vol. 2012, pp. 1–7. URL: https://cw.fel.cvut.cz/b241/_media/courses/b4m39rso/lectures/s2012_pbs_disney_brdf_notes_v3.pdf (visited on 06/21/2025).

Chaitanya, C. R. A., A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila (Aug. 31, 2017). "Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder". In: *ACM Transactions on Graphics* 36.4, pp. 1–12. URL: https://dl.acm.org/doi/10.1145/3072959.3073601 (visited on 08/25/2025).

Christensen, P. H. and W. Jarosz (2016). "The Path to Path-Traced Movies". In: *Foundations and Trends® in Computer Graphics and Vision* 10.2, pp. 103–175. URL: https://graphics.pixar.com/library/PathTracedMovies/paper.pdf (visited on 08/11/2025).

Cigolle, Z. H., S. Donow, D. Evangelakos, M. Mara, M. McGuire, and Q. Meyer (2014). "A Survey of Efficient Representations for Independent Unit Vectors". In: *Journal of Computer Graphics Techniques* 3.2. URL: https://jcgt.org/published/0003/02/01/paper-lowres.pdf.

Cook, R. L. and K. E. Torrance (Jan. 1982). "A Reflectance Model for Computer Graphics". In: *ACM Transactions on Graphics* 1.1, pp. 7–24. URL: https://dl.acm.org/doi/10.1145/357290.357293 (visited on 02/23/2024).

Cranley, R. and T. N. L. Patterson (1976). "Randomization of Number Theoretic Methods for Multiple Integration". In: *SIAM Journal on Numerical Analysis* 13.6, pp. 904–914. JSTOR: 2156452. URL: https://www.jstor.org/stable/2156452 (visited on 06/15/2025).

Dahm, K. and A. Keller (July 30, 2017). "Learning Light Transport the Reinforced Way". In: *ACM SIGGRAPH 2017 Talks*. SIGGRAPH '17: Special Interest Group on Computer Graphics and Interactive Techniques Conference. Los Angeles California: ACM, pp. 1–2. URL: https://dl.acm.org/doi/10.1145/3084363.3085032 (visited on 08/22/2025).

Dereviannykh, M., D. Klepikov, J. Hanika, and C. Dachsbacher (Dec. 5, 2024). *Neural Two-Level Monte Carlo Real-Time Rendering*. Version 1. arXiv: 2412.04634 [cs]. URL: http://arxiv.org/abs/2412.04634 (visited on 05/10/2025). Pre-published.

Dupuy, J. and A. Benyoub (June 12, 2023). *Sampling Visible GGX Normals with Spherical Caps*. arXiv: 2306.05044 [cs]. URL: http://arxiv.org/abs/2306.05044 (visited on 08/28/2024). Pre-published.

Engelhardt, T. and C. Dachsbacher (2008). "Octahedron Environment Maps." In: *VMV*. Citeseer, pp. 383–388. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fcb9a6dbdf7b4c31f94e481cf101c83b73ea6410 (visited on 12/21/2024).

Evangelou, I., G. Papaioannou, K. Vardis, and A. Vasilakis (Feb. 5, 2021). "Fast Radius Search Exploiting Ray-Tracing Frameworks, JCGT". In: 10, p. 2021. URL: https://jcgt.org/published/0010/01/02/.

Fujii, Y. (2025). *A Tiny Improvement of Oren-Nayar Reflectance Model*. URL: https://mimosa-pudica.net/improved-oren-nayar.html (visited on 06/14/2025).

Georgiev, I., J. Křivánek, T. Davidovič, and P. Slusallek (Nov. 2012). "Light Transport Simulation with Vertex Connection and Merging". In: *ACM Transactions on Graphics* 31.6, pp. 1–10. URL: https://dl.acm.org/doi/10.1145/2366145.2366211 (visited on 05/18/2025).

Goral, C. M., K. E. Torrance, D. P. Greenberg, and B. Battaile (July 1984). "Modeling the Interaction of Light between Diffuse Surfaces". In: *ACM SIGGRAPH Computer Graphics* 18.3, pp. 213–222. URL: https://dl.acm.org/doi/10.1145/964965.808601 (visited on 08/11/2025).

Green, R. (Jan. 16, 2003). "Spherical Harmonic Lighting: The Gritty Details". In: *Archives of the Game Developers Conference*. Vol. 56, p. 4. URL: https://3dvar.com/Green2003Spherical.pdf (visited on 08/22/2025).

Greger, G., P. Shirley, P. M. Hubbard, and D. P. Greenberg (Mar.–Apr. 1998). "The Irradiance Volume". In: *IEEE Computer Graphics and Applications* 18.2, pp. 32–43. URL: https://ieeexplore.ieee.org/abstract/document/656788/ (visited on 06/27/2025).

Hachisuka, T. and H. W. Jensen (Dec. 1, 2009). "Stochastic Progressive Photon Mapping". In: *ACM SIGGRAPH Asia 2009 Papers*. SIGGRAPH Asia '09. New York, NY, USA: Association for Computing Machinery, pp. 1–8. URL: https://dl.acm.org/doi/10.1145/1661412.1618487 (visited on 05/13/2025).

— (Dec. 15, 2010). "Parallel Progressive Photon Mapping on GPUs". In: *ACM SIGGRAPH ASIA 2010 Sketches*. SA '10. New York, NY, USA: Association for Computing Machinery, p. 1. URL: https://dl.acm.org/doi/10.1145/1899950.1900004 (visited on 05/13/2025).

Hachisuka, T., S. Ogaki, and H. W. Jensen (Dec. 1, 2008). "Progressive Photon Mapping". In: *ACM Trans. Graph.* 27.5, 130:1–130:8. URL: https://dl.acm.org/doi/10.1145/1409060.1409083 (visited on 05/13/2025).

Halén, H., K. Hayward, A. Brinck, and X. Bei (2021). "Global Illumination Based on Surfels". In: *ACM SIGGRAPH 2021 Advances in Real-Time Rendering in Game Course*. SIGGRAPH 2021. URL: https://advances.realtimerendering.com/s2021/index.html (visited on 08/23/2025).

Hasselgren, J., J. Munkberg, M. Salvi, A. Patney, and A. Lefohn (May 2020). "Neural Temporal Adaptive Sampling and Denoising". In: *Computer Graphics Forum* 39.2, pp. 147–155. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13919 (visited on 09/03/2025).

Heckbert, P. S. (Sept. 1990). "Adaptive Radiosity Textures for Bidirectional Ray Tracing". In: *ACM SIGGRAPH Computer Graphics* 24.4, pp. 145–154. URL: https://dl.acm.org/doi/10.1145/97880.97895 (visited on 04/24/2025).

Hedstrom, T., M. Kettunen, D. Lin, C. Wyman, and T.-M. Li (June 17, 2025). "ReSTIR BDPT: Bidirectional ReSTIR Path Tracing with Caustics". In: *ACM Trans. Graph.* URL: https://dl.acm.org/doi/10.1145/3744898 (visited on 07/10/2025).

Heinrich, S. and A. Keller (1994). *Quasi-Monte Carlo Methods in Computer Graphics, Part I: The QMC-Bu Er*. Citeseer. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f9a420d0c5739e48b82f4eb7ab9e017f4e7eb658 (visited on 06/21/2025).

Heitz, E. (2014). "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs". In: *Journal of Computer Graphics Techniques* 3.2, pp. 32–91. URL: https://inria.hal.science/hal-01024289/ (visited on 08/21/2025).

— (2018). "Sampling the GGX Distribution of Visible Normals". In: *Journal of Computer Graphics Techniques* 7.4.

Hermosilla, P., S. Maisch, T. Ritschel, and T. Ropinski (June 30, 2019). *Deep-Learning the Latent Space of Light Transport*. arXiv: 1811.04756 [cs]. URL: http://arxiv.org/abs/1811.04756 (visited on 08/25/2025). Pre-published.

Immel, D. S., M. F. Cohen, and D. P. Greenberg (Aug. 31, 1986). "A Radiosity Method for Non-Diffuse Environments". In: *ACM SIGGRAPH Computer Graphics* 20.4, pp. 133–142. URL: https://dl.acm.org/doi/10.1145/15886.15901 (visited on 08/21/2025).

Iwanicki, M. and P.-P. Sloan (2017). "Precomputed Lighting in Call of Duty: Infinite Warfare". In: *Advances in Real-Time Rendering in Games, Part I (ACM SIGGRAPH Courses)*. URL: https://advances.realtimerendering.com/s2017/.

Jakob, W. and S. Marschner (Aug. 5, 2012). "Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport". In: *ACM Transactions on Graphics* 31.4, pp. 1–13. URL: https://dl.acm.org/doi/10.1145/2185520.2185554 (visited on 08/21/2025).

Jensen, H. W. (1995). "Importance Driven Path Tracing Using the Photon Map". In: *Rendering Techniques '95*. Ed. by P. M. Hanrahan and W. Purgathofer. Vienna: Springer Vienna, pp. 326–335. URL: http://link.springer.com/10.1007/978-3-7091-9430-0_31 (visited on 06/25/2025).

— (1996). "Global Illumination Using Photon Maps". In: *Rendering Techniques '96*. Ed. by X. Pueyo and P. Schröder. Vienna: Springer Vienna, pp. 21–30. URL: http://link.springer.com/10.1007/978-3-7091-7484-5_3 (visited on 04/22/2025).

Jiang, G. and B. Kainz (July 30, 2020). *Deep Radiance Caching: Convolutional Autoencoders Deeper in Ray Tracing*. arXiv: 1910.02480 [cs]. URL: http://arxiv.org/abs/1910.02480 (visited on 08/25/2025). Pre-published.

Kajiya, J. T. (Aug. 31, 1986). "The Rendering Equation". In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, pp. 143–150. URL: https://dl.acm.org/doi/10.1145/15922.15902 (visited on 02/23/2024).

Kang, C.-m., L. Wang, Y.-n. Xu, and X.-x. Meng (Mar. 1, 2016). "A Survey of Photon Mapping State-of-the-Art Research and Future Challenges". In: *Frontiers of Information Technology & Electronic Engineering* 17.3, pp. 185–199. URL: https://doi.org/10.1631/FITEE.1500251 (visited on 05/18/2025).

Karis, B. and Epic Games (2013). "Real Shading in Unreal Engine 4". In: *Proc. Physically Based Shading Theory Practice* 4.3, p. 1. URL: https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf (visited on 09/10/2025).

Kautz, J., J. Snyder, and P.-P. J. Sloan (2002). "Fast Arbitrary Brdf Shading for Low-Frequency Lighting Using Spherical Harmonics." In: *Rendering Techniques* 2.291–296, p. 1. URL: https://www.ppsloan.org/publications/shbrdf_final17.pdf (visited on 08/22/2025).

Keller, A. (1995). "A Quasi-Monte Carlo Algorithm for the Global Illumination Problem in the Radiosity Setting". In: *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. Ed. by H. Niederreiter and P. J.-S. Shiue. Red. by P. Bickel, P. Diggle, S. Fienberg, K. Krickeberg, I. Olkin, N. Wermuth, and S. Zeger. Vol. 106. New York, NY: Springer New York, pp. 239–251. URL: http://link.springer.com/10.1007/978-1-4612-2552-2_15 (visited on 06/21/2025).

Keller, A. (1996). "Quasi-Monte Carlo Radiosity". In: *Rendering Techniques '96*. Ed. by X. Pueyo and P. Schröder. Vienna: Springer Vienna, pp. 101–110. URL: http://link.springer.com/10.1007/978-3-7091-7484-5_11 (visited on 06/21/2025).

— (1997). "Instant Radiosity". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97*. The 24th Annual Conference. Not Known: ACM Press, pp. 49–56. URL: http://portal.acm.org/citation.cfm?doid=258734.258769 (visited on 08/25/2025).

Keller, A., K. Dahm, and N. Binder (2016). "Path Space Filtering". In: *Monte Carlo and Quasi-Monte Carlo Methods*. Ed. by R. Cools and D. Nuyens. Vol. 163. Cham: Springer International Publishing, pp. 423–436. URL: http://link.springer.com/10.1007/978-3-319-33507-0_21.

Kern, R., F. Brüll, and T. Grosch (2023). "Accelerating Photon Mapping for Hardware-Based Ray Tracing". In: *Journal of Computer Graphics Techniques Vol* 12.1. URL: https://jcgt.org/published/0012/01/01/paper-lowres.pdf (visited on 06/21/2025).

— (2024). *ReSTIR FG: Real-Time Reservoir Resampled Photon Final Gathering*. The Eurographics Association. URL: https://doi.org/10.2312/sr.20241155 (visited on 05/13/2025).

Kingma, D. and J. Ba (Dec. 22, 2014). *Adam: A Method for Stochastic Optimization*. Version 1. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 09/07/2025). Pre-published.

Kirkpatrick, A., J. O Connor, A. Campbell, and M. Cooper (May 6, 2025). *Web Content Accessibility Guidelines (WCAG) 2.1*. World Wide Web Consortium. URL: https://www.w3.org/TR/WCAG21/#dfn-relative-luminance (visited on 06/21/2025).

Knaus, C. and M. Zwicker (May 2011). "Progressive Photon Mapping: A Probabilistic Approach". In: *ACM Transactions on Graphics* 30.3, pp. 1–13. URL: https://dl.acm.org/doi/10.1145/1966394.1966404 (visited on 07/12/2025).

Křivánek, J., P. Gautron, S. Pattanaik, and K. Bouatouch (2005). "Radiance Caching for Efficient Global Illumination Computation". In: *IEEE Transactions on Visualization and Computer Graphics* 11.5, pp. 550–561. URL: https://ieeexplore.ieee.org/abstract/document/1471692/ (visited on 07/28/2025).

Křivánek, J., P. Gautron, G. Ward, O. Arikan, and H. W. Jensen (Aug. 5, 2007). "Practical Global Illumination with Irradiance Caching". In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH07: Special Interest Group on Computer Graphics and Interactive Techniques Conference. San Diego California: ACM, p. 1. URL: https://dl.acm.org/doi/10.1145/1281500.1281617 (visited on 08/22/2025).

Kuenlin, Q. (Mar. 18, 2024). "Advanced Graphics Summit: Raytracing in Snowdrop: An Optimized Lighting Pipeline for Consoles". In: *Advanced Graphics Summit*. Game Developers Conference 2024. Massive Entertainment a Ubisoft Studio. URL: https://uat.gdcvault.com/play/1034763/Advanced-Graphics-Summit-Raytracing-in (visited on 08/24/2025).

Lafortune, E. P. and Y. D. Willems (1993). "Bi-Directional Path Tracing". In: URL: http://masters.donntu.ru/2013/fknt/kalamitra/library/Bidirectional%20Path%20Tracing%201990.pdf (visited on 08/11/2025).

— (1995). "A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing". In: *Rendering Techniques '95*. Ed. by P. M. Hanrahan and W. Purgathofer. Vienna: Springer Vienna, pp. 11–20. URL: http://link.springer.com/10.1007/978-3-7091-9430-0_2 (visited on 06/25/2025).

Lagarde, S. and C. de Rousiers (2014). "Moving Frostbite to Physically Based Rendering 3.0". In: *SIGGRAPH*.

Lehtinen, J., J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila (Oct. 29, 2018). *Noise2Noise: Learning Image Restoration without Clean Data*. arXiv: 1803.04189 [cs]. URL: http://arxiv.org/abs/1803.04189 (visited on 08/02/2025). Pre-published.

Lin, D., M. Kettunen, B. Bitterli, J. Pantaleoni, C. Yuksel, and C. Wyman (July 2022). "Generalized Resampled Importance Sampling: Foundations of ReSTIR". In: *ACM Transactions on Graphics* 41.4, pp. 1–23. URL: https://dl.acm.org/doi/10.1145/3528223.3530158 (visited on 04/30/2025).

Majercik, Z., J.-P. Guertin, D. Nowrouzezahrai, and M. McGuire (2019). "Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields". In: *Journal of Computer Graphics Techniques* 8.2. URL: https://cim.mcgill.ca/~derek/files/DDGI-lowres.pdf (visited on 07/29/2025).

Majercik, Z., T. Müller, A. Keller, D. Nowrouzezahrai, and M. McGuire (July 31, 2021). "Dynamic Diffuse Global Illumination Resampling". In: *ACM SIGGRAPH 2021 Talks*. SIGGRAPH '21: Special Interest Group on Computer Graphics and Interactive Techniques Conference. Virtual Event USA: ACM, pp. 1–2. URL: https://dl.acm.org/doi/10.1145/3450623.3464635 (visited on 07/29/2025).

Mara, M., D. Luebke, and M. McGuire (Mar. 21, 2013). "Toward Practical Real-Time Photon Mapping: Efficient GPU Density Estimation". In: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '13. New York, NY, USA: Association for Computing Machinery, pp. 71–78. URL: https://dl.acm.org/doi/10.1145/2448196.2448207 (visited on 05/17/2025).

McGuire, M., M. Mara, D. Nowrouzezahrai, and D. Luebke (Feb. 25, 2017). "Real-Time Global Illumination Using Precomputed Light Field Probes". In: *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '17: Symposium on Interactive 3D Graphics and Games. San Francisco California: ACM, pp. 1–11. URL: https://dl.acm.org/doi/10.1145/3023368.3023378 (visited on 08/22/2025).

Metropolis, N. and S. Ulam (Sept. 1949). "The Monte Carlo Method". In: *Journal of the American Statistical Association* 44.247, pp. 335–341. URL: http://www.tandfonline.com/doi/abs/10.1080/01621459.1949.10483310 (visited on 08/08/2025).

Mildenhall, B., P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng (Aug. 3, 2020). *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. arXiv: 2003.08934 [cs]. URL: http://arxiv.org/abs/2003.08934 (visited on 08/25/2025). Pre-published.

Moreau, P., M. Pharr, and P. Clarberg (2019). "Dynamic Many-Light Sampling for Real-Time Ray Tracing". In: *High Performance Graphics (Short Papers)*, pp. 21–26. URL: https://research.nvidia.com/sites/default/files/pubs/2019-07_Dynamic-Many-Light-Sampling//MPC19.pdf.

Müller, T. (Apr. 2021). *Tiny-Cuda-Nn*. Version 2.0. URL: https://github.com/NVlabs/tiny-cuda-nn (visited on 09/07/2025).

Müller, T., A. Evans, C. Schied, and A. Keller (July 2022). "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Transactions on Graphics* 41.4, pp. 1–15. URL: https://dl.acm.org/doi/10.1145/3528223.3530127 (visited on 04/22/2025).

Müller, T., M. Gross, and J. Novák (July 2017). "Practical Path Guiding for Efficient Light-Transport Simulation". In: *Computer Graphics Forum* 36.4, pp. 91–100. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13227 (visited on 05/16/2025).

Müller, T., B. McWilliams, F. Rousselle, M. Gross, and J. Novák (Sept. 3, 2019). *Neural Importance Sampling*. arXiv: 1808.03856 [cs]. URL: http://arxiv.org/abs/1808.03856 (visited on 04/23/2025). Pre-published.

Müller, T., F. Rousselle, J. Novák, and A. Keller (Sept. 4, 2020). *Neural Control Variates*. arXiv: 2006.01524 [cs]. URL: http://arxiv.org/abs/2006.01524 (visited on 07/06/2025). Pre-published.

— (Aug. 31, 2021). "Real-Time Neural Radiance Caching for Path Tracing". In: *ACM Transactions on Graphics* 40.4, pp. 1–16. URL: https://dl.acm.org/doi/10.1145/3450626.3459812 (visited on 09/24/2024).

Nalbach, O., E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel (July 2017). "Deep Shading: Convolutional Neural Networks for Screen Space Shading". In: *Computer Graphics Forum*

36.4, pp. 65–78. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13225 (visited on 08/25/2025).

NVIDIA Corporation (Jan. 2024). *cuRAND Library Programming Guide*. URL: https://docs.nvidia.com/cuda/pdf/CURAND_Library.pdf.

O'Donnell, Y. (2018). "Precomputed Global Illumination in Frostbite". In: *Game Developers Conference*. Vol. 1. 2.

Oren, M. and S. K. Nayar (1994). "Generalization of Lambert's Reflectance Model". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '94*. The 21st Annual Conference. Not Known: ACM Press, pp. 239–246. URL: http://portal.acm.org/citation.cfm?doid=192161.192213 (visited on 06/24/2025).

Ouyang, Y., S. Liu, M. Kettunen, M. Pharr, and J. Pantaleoni (Dec. 2021). "ReSTIR GI: Path Resampling for Real-Time Path Tracing". In: *Computer Graphics Forum* 40.8, pp. 17–29. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.14378 (visited on 04/22/2025).

Owen, A. B. (1995). "Randomly Permuted (t,m,s)-Nets and (t, s)-Sequences". In: *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. Ed. by H. Niederreiter and P. J.-S. Shiue. New York, NY: Springer, pp. 299–317.

— (Oct. 1, 2008). "Local Antithetic Sampling with Scrambled Nets". In: *The Annals of Statistics* 36.5. arXiv: 0811.0528 [stat]. URL: http://arxiv.org/abs/0811.0528 (visited on 06/15/2025).

Poly Haven (2025). *Poly Haven*. Poly Haven. URL: https://polyhaven.com/ (visited on 09/06/2025).

Ren, P., J. Wang, M. Gong, S. Lin, X. Tong, and B. Guo (2013). "Global Illumination with Radiance Regression Functions". In: *ACM Trans. Graph.* 32.4, pp. 130–1. URL: https://www.academia.edu/download/54395664/Global_illumination_with_radiance_regres20170910-8835-10klmux.pdf (visited on 08/25/2025).

Ritschel, T., C. Dachsbacher, T. Grosch, and J. Kautz (Feb. 2012). "The State of the Art in Interactive Global Illumination". In: *Computer Graphics Forum* 31.1, pp. 160–188. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2012.02093.x (visited on 05/04/2025).

Scan the World (Aug. 18, 2014). *The Thinker (Auguste Rodin). Licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/)*. URL: https://commons.wikimedia.org/wiki/File:Scan_the_World_-_The_Thinker_(Auguste_Rodin).stl (visited on 09/06/2025).

Schied, C., A. Kaplanyan, C. Wyman, A. Patney, C. R. A. Chaitanya, J. Burgess, S. Liu, C. Dachsbacher, A. Lefohn, and M. Salvi (July 28, 2017). "Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination". In: *Proceedings of High Performance Graphics*. HPG '17: High-Performance Graphics. Los Angeles California: ACM, pp. 1–12. URL: https://dl.acm.org/doi/10.1145/3105762.3105770 (visited on 08/25/2025).

Schlick, C. (Aug. 1994). "An Inexpensive BRDF Model for Physically-based Rendering". In: *Computer Graphics Forum* 13.3, pp. 233–246. URL: https://onlinelibrary.wiley.com/doi/10.1111/1467-8659.1330233 (visited on 06/24/2025).

Sloan, P.-P., J. Kautz, and J. Snyder (July 1, 2002). "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments". In: *ACM Trans. Graph.* 21.3. Ed. by M. C. Whitton, pp. 527–536. URL: https://www.ppsloan.org/publications/shillum_final23.pdf (visited on 08/22/2025).

Smith, B. G. (Aug. 15, 1967). "Lunar Surface Roughness: Shadowing and Thermal Emission". In: *Journal of Geophysical Research* 72.16, pp. 4059–4067. URL: http://doi.wiley.com/10.1029/JZ072i016p04059 (visited on 06/24/2025).

Stachowiak, T. (July 25, 2018). "Stochastic All the Things: Raytracing in Hybrid Real-Time Rendering". In: *Programming and Technology Track*. Digital Dragons 2018. Electronic Arts Inc. URL: https://www.ea.com/seed/news/seed-dd18-presentation-slides-raytracing (visited on 08/24/2025).

Stamm, J. C. (Aug. 5, 2025). *Julcst/Binrc*. URL: https://github.com/julcst/binrc.

Stanford University Computer Graphics Laboratory (1994). *Stanford Bunny*. URL: https://graphics.stanford.edu/data/3Dscanrep/ (visited on 03/11/2024).

Talbot, J. F. (2005). "Importance Resampling for Global Illumination". In: URL: https://scholar.archive.org/work/sxqgndnsanea5ffxwmopgy42uy/access/wayback/https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=1662&context=etd (visited on 08/24/2025).

Tancik, M., P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng (2020). "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains". In: *Advances in neural information processing systems* 33, pp. 7537–7547. URL: https://proceedings.neurips.cc/paper_files/paper/2020/hash/55053683268957697aa39fba6f231c68-Abstract.html (visited on 07/07/2025).

Tatzgern, W., A. Weinrauch, P. Stadlbauer, J. H. Mueller, M. Winter, and M. Steinberger (Aug. 9, 2024). "Radiance Caching with On-Surface Caches for Real-Time Global Illumination". In:

Proceedings of the ACM on Computer Graphics and Interactive Techniques 7.3, pp. 1–17. URL: https://dl.acm.org/doi/10.1145/3675382 (visited on 06/28/2025).

TensorFlow Developers (May 2021). *TensorFlow*. Version v2.5.0. Zenodo. URL: https://doi.org/10.5281/zenodo.4758419.

The Khronos® 3D Formats Working Group (Oct. 11, 2021). *glTF™ 2.0 Specification*. URL: https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html.

Thomas, M. M. and A. G. Forbes (May 22, 2018). *Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network*. arXiv: 1710.09834 [cs]. URL: http://arxiv.org/abs/1710.09834 (visited on 07/29/2025). Pre-published.

Tole, P., F. Pellacini, B. Walter, and D. P. Greenberg (July 2002). "Interactive Global Illumination in Dynamic Scenes". In: *ACM Transactions on Graphics* 21.3, pp. 537–546. URL: https://dl.acm.org/doi/10.1145/566654.566613 (visited on 08/22/2025).

Torrance, K. E. and E. M. Sparrow (Sept. 1, 1967). "Theory for Off-Specular Reflection From Roughened Surfaces*". In: *Journal of the Optical Society of America* 57.9, pp. 1105–1114. URL: https://opg.optica.org/abstract.cfm?URI=josa-57-9-1105 (visited on 06/25/2025).

Trowbridge, T. S. and K. P. Reitz (May 1, 1975). "Average Irregularity Representation of a Rough Surface for Ray Reflection". In: *Journal of the Optical Society of America* 65.5, p. 531. URL: https://opg.optica.org/abstract.cfm?URI=josa-65-5-531 (visited on 03/09/2024).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention Is All You Need". In: *Advances in neural information processing systems* 30. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 07/08/2025).

Veach, E. (Dec. 1997). "Robust Monte Carlo Methods for Light Transport Simulation". Stanford University. URL: https://graphics.stanford.edu/papers/veach_thesis/thesis.pdf (visited on 12/21/2024).

Veach, E. and L. J. Guibas (1997). "Metropolis Light Transport". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97*. The 24th Annual Conference. Not Known: ACM Press, pp. 65–76. URL: http://portal.acm.org/citation.cfm?doid=258734.258775 (visited on 08/21/2025).

Vorba, J., O. Karlík, M. Šik, T. Ritschel, and J. Křivánek (July 27, 2014). "On-Line Learning of Parametric Mixture Models for Light Transport Simulation". In: *ACM Transactions on Graphics* 33.4, 101:1–101:11. URL: https://dl.acm.org/doi/10.1145/2601097.2601203 (visited on 08/21/2025).

Walter, B., G. Drettakis, and S. Parker (1999). "Interactive Rendering Using the Render Cache". In: *Rendering Techniques' 99*. Ed. by D. Lischinski and G. W. Larson. Vienna: Springer Vienna, pp. 19–30. URL: http://link.springer.com/10.1007/978-3-7091-6809-7_3 (visited on 08/22/2025).

Walter, B., S. R. Marschner, H. Li, and K. E. Torrance (2007). "Microfacet Models for Refraction through Rough Surfaces." In: *Rendering techniques* 2007, 18th. URL: https://www.graphics.cornell.edu/~bjw/microfacetbsdf.pdf (visited on 06/25/2025).

Ward, G. J., F. M. Rubinstein, and R. D. Clear (Aug. 1988). "A Ray Tracing Solution for Diffuse Interreflection". In: *ACM SIGGRAPH Computer Graphics* 22.4, pp. 85–92. URL: https://dl.acm.org/doi/10.1145/378456.378490 (visited on 04/22/2025).

Whitted, T. (1980). "An Improved Illumination Model for Shaded Display". In: *Communications of the ACM* 23.6, pp. 343–349. URL: https://cir.nii.ac.jp/crid/1362544420076304768 (visited on 08/11/2025).

Wright, D. (2021). "Radiance Caching for Real-Time Global Illumination". In: *ACM SIGGRAPH 2021 Advances in Real-Time Rendering in Game Course*, pp. 9–13. URL: https://advances.realtimerendering.com/s2021/index.html.

Wright, D., K. Narkowicz, and P. Kelly (Aug. 2022). "Real-Time Global Illumination in Unreal Engine 5". In: *SIGGRAPH 2022 Advances in Real-Time Rendering in Games Course*. SIGGRAPH 2022. Vancouver. URL: https://advances.realtimerendering.com/s2022/SIGGRAPH2022-Advances-Lumen-Wright%20et%20al.pdf (visited on 06/28/2025).

Wyman, C., M. Kettunen, D. Lin, B. Bitterli, C. Yuksel, W. Jarosz, and P. Kozlowski (July 24, 2023). "A Gentle Introduction to ReSTIR Path Reuse in Real-Time". In: *ACM SIGGRAPH 2023 Courses*. SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery, pp. 1–38. URL: https://dl.acm.org/doi/10.1145/3587423.3595511 (visited on 07/11/2025).

Zhu, S., Z. Xu, H. W. Jensen, H. Su, and R. Ramamoorthi (Apr. 25, 2020). *Deep Photon Mapping*. arXiv: 2004.12069 [cs]. URL: http://arxiv.org/abs/2004.12069 (visited on 04/30/2025). Pre-published.