DOI: 10.1111/cgf.70050
EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

Neural Two-Level Monte Carlo Real-Time Rendering

Mikhail Dereviannykh Dmitrii Klepikov Johannes Hanika Carsten Dachsbacher

Karlsruhe Institute of Technology

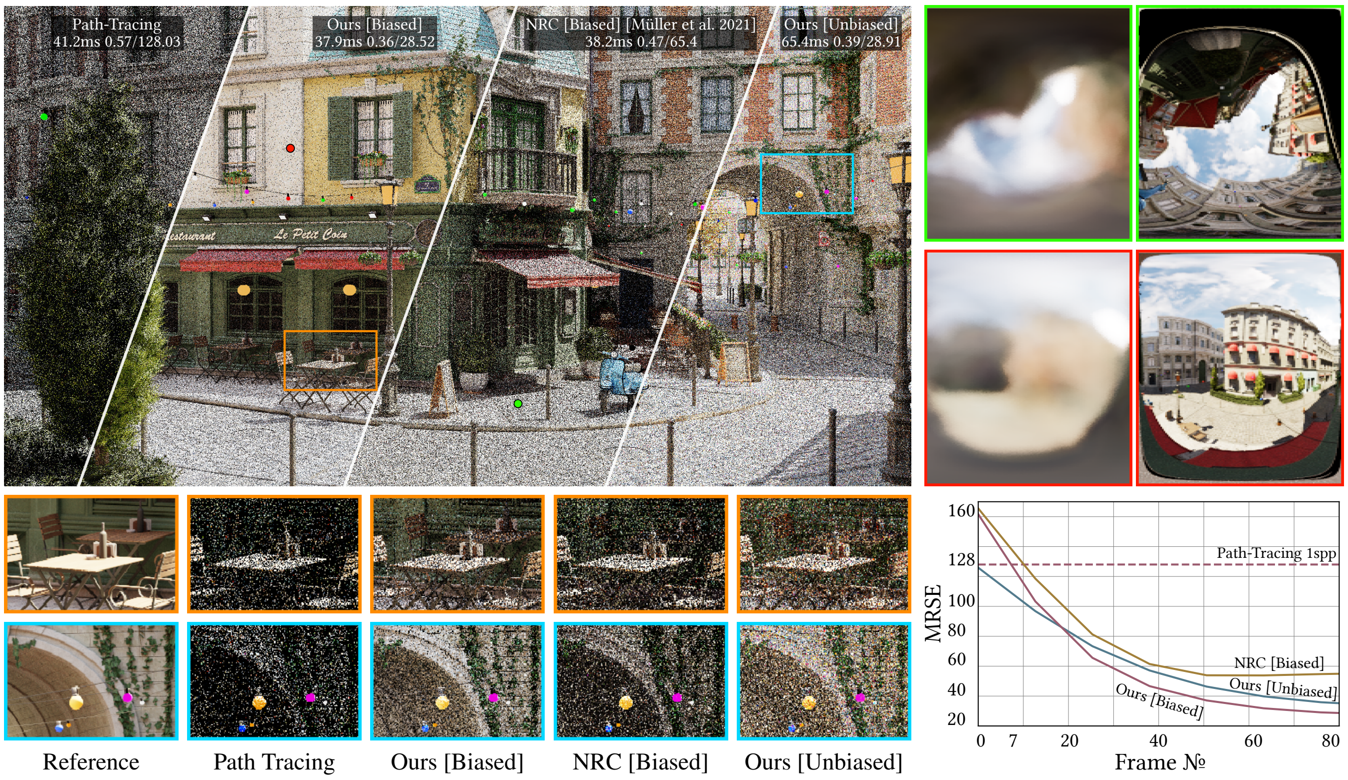


Figure 1: *The Bistro scene (RTX 3080 at 1080p) rendered using 1 spp to compare our* Neural Incident Radiance Cache *(NIRC) to path tracing and the Neural Radiance Cache (NRC) [MRNK21] with a static camera. As does NRC, we replace tracing long paths with a cache lookup. NIRC, however, stores incident radiance, avoiding a ray cast before lookup. In addition, combining* Two-Level Monte Carlo *with our NIRC enables us to estimate the cache error and thus to remove bias. For each render, we show compute time, the image difference (ꟻLIP, MRSE). On the right, the top row shows NIRC cache visualizations (left) vs reference (right) with a convergence plot below.*

Abstract

We introduce an efficient Two-Level Monte Carlo (subset of Multi-Level Monte Carlo, MLMC) estimator for real-time rendering of scenes with global illumination. Using MLMC we split the shading integral into two parts: the radiance cache integral and the residual error integral that compensates for the bias of the first one. For the first part, we developed the Neural Incident Radiance Cache *(NIRC) leveraging the power of tiny neural networks [MRNK21] as a building block, which is trained on the fly. The cache is designed to provide a fast and reasonable approximation of the incident radiance: an evaluation takes 2-25× less compute time than a path tracing sample. This enables us to estimate the radiance cache integral with a high number of samples and by this achieve faster convergence. For the residual error integral, we compute the difference between the NIRC predictions and the unbiased path tracing simulation. Our method makes no assumptions about the geometry, materials, or lighting of a scene and has only few intuitive hyper-parameters. We provide a comprehensive comparative analysis in different experimental scenarios. Since the algorithm is trained in an on-line fashion, it demonstrates significant noise level reduction even for dynamic scenes and can easily be combined with other noise reduction techniques.*

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Ray tracing; Neural networks;***

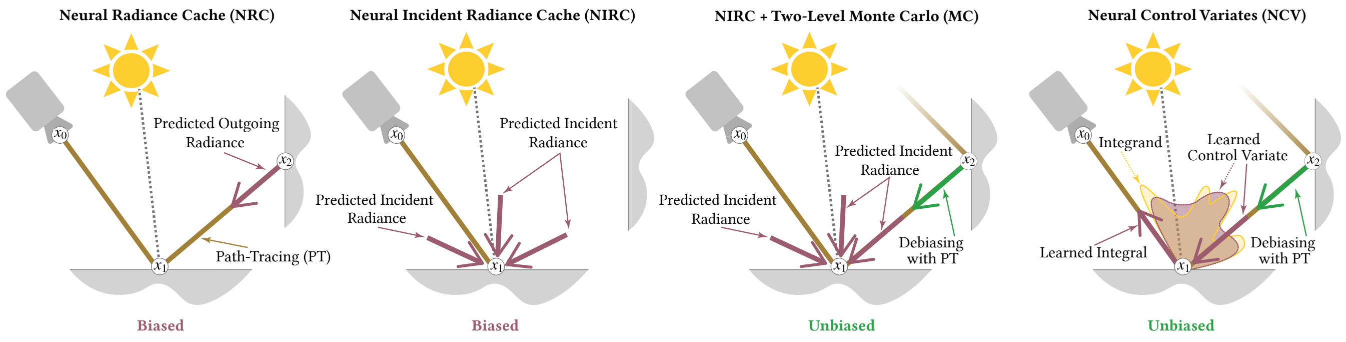

Neural Radiance Cache (NRC)
Neural Incident Radiance Cache (NIRC)
NIRC + Two-Level Monte Carlo (MC)
Neural Control Variates (NCV)
x_0
x_1
x_2
Predicted Outgoing Radiance
Path-Tracing (PT)
Predicted Incident Radiance
Predicted Incident Radiance
Predicted Incident Radiance
Predicted Incident Radiance
Debiasing with PT
Integrand
Learned Control Variate
Learned Integral
Debiasing with PT
Biased
Biased
Unbiased
Unbiased

Figure 2: *This figure illustrates the application of the* Neural Radiance Cache *(NRC) [MRNK21] and our* Neural Incident Radiance Cache *(NIRC) in biased path tracing on the left side. In NRC, a path is traced from the camera x_0 to surface point x_1, where it scatters and then terminates at x_2 in the cache. In contrast, NIRC stops tracing already at x_1 and estimates radiance using* Monte Carlo *(MC), as described in Equation (12), by sampling scattering rays based on BRDF and querying the NIRC to predict incident radiance. In practice, the exact termination point is determined by different heuristics for both models (see Section 4.2). On the right side, the figure shows the application of the* NIRC *combined with* Two-Level MC *and* Neural Control Variates *(NCV) [MRKN20] for the unbiased rendering. In both cases, the render continues* Path-Tracing *(PT) and estimates residual errors using MC - "debiasing". NCV trains Normalizing Flows [MMR*19] to act as a control variate, allowing analytical estimation of the rendering integral. As the NIRC estimates the integral stochastically, it alleviates neural architectural constraints and leads to better variance reduction compared to NCV in our experiments (see Section 6.1).*

1. Introduction

Rendering global illumination effects remains challenging in interactive and real-time applications, even with modern hardware-accelerated ray tracing. Monte Carlo methods, which are used to compute a solution to the *Rendering Equation* [Kaj86] are prone to noise due to their stochastic nature, amplified by complex materials and occlusions in a scene. Techniques such as importance sampling, and more recently learning-based approaches have demonstrated effectiveness in addressing these issues, mitigating the noise problem significantly using radiance caching [GKB09; WRC88; BMdD*23], photon mapping [Jen96; HOJ08], adaptive sampling [Mit87], vertex connection and merging [GKDS12] and path guiding [VHH*19].

Our work also depends on caching: we use several path tracing samples to initialize a cached representation of the incident radiance field in the scene. This cache can then be evaluated to yield a fast approximation of the actual light field.
In our approach, we introduce a novel caching method using fully-fused tiny neural networks, inspired by the *Neural Radiance Cache* (NRC) [MRNK21], and multiresolution hash encoding [MESK22]. The important novelty of our method is that we employ *Neural Incident Radiance Caches* (NIRCs) which are specifically designed to cache incident radiance (as opposed to outgoing radiance with NRCs). This allows us to query the cache for incident radiance at a shading point post-material evaluation to efficiently approximate the shading integral without additional ray tracing (see Figure 2). Our combination with a *Two-Level Monte Carlo* scheme enables us to compensate for the resulting bias (see Figure 2). For biased rendering, we introduce a new Balanced Termination Heuristic (BTH) to enhance path termination efficiency, leveraging the strengths of NIRCs while addressing their limitations with glossy details. The BTH allows us to use our cache instantly at the primary bounce since it predicts incident radiance directly, unlike the NRC *Spread Angle Heuristic* (SPH), which is unsuitable for stopping at the primarily visible surface.

Along with that, we assessed the differences between *Control Variates* (CV) and Two-Level Monte Carlo methods by comparing analytical models, such as *Spherical Harmonics* (SH) and mixtures of *von Mises-Fisher* (vMF) lobes, commonly used in the earlier research [Pan20][VKŠ*14], alongside *Neural Control Variates* (NCV) [MRKN20]. While NCV is specifically designed to satisfy Control Variates requirements by providing analytical integral computation (see Figure 2), our results show that NIRC in the Two-Level Monte Carlo framework can achieve lower variance of the residual error estimator, requiring significantly fewer training frames and offering higher generalization.

Furthermore, the NIRC is designed to predict incident radiance for batches of 10 to 50 rays, amortizing the computation of the spatial feature for requests for the same shading point. This can lead to significant speedups compared to NRC. We also introduce an alternative encoding of the input to the multi-layer perceptron (MLP) [Hay94], leading to a much more precise angular representation of the radiance field which is required for sufficiently accurate approximation of the shading integral. To train the cache, we derive a loss function which includes the two-level Monte Carlo estimator; further, we also evaluate a simplified loss based on the relative L2 difference.

Additionally, we developed an error-based path termination algorithm to ensure a fair equal-bias comparison between the NIRC and the NRC. Our cache analysis shows that NIRC can achieve 2.3–11.2x fewer indirect radiance bounces compared to NRC, performing them only for 5–40% of the pixels while maintaining similar bias levels and meeting reasonable rendering quality standards. In extreme cases, when the quality requirements are significantly relaxed, the reduction of indirect radiance bounces can reach 2 orders of magnitude, with tracing required for less than 1% of the pixels. Using the SPH, NIRC achieves a Relative Squared Bias up to 4.12× lower than NRC, though it increases noise. If we accept higher bias, BTH reduces indirect radiance bounces by up to 3.54 times while slightly decreasing the *Mean Relative Squared Error* (MRSE), enhancing real-time rendering performance and accuracy. These tradeoffs are analyzed and discussed in more detail in Section 6.3.

Finally, we investigate the specific use case of environment map lighting. Instead of approximating the radiance [RFGL23], the neural cache is used to predict the visibility term, which is similar to a concurrent work [FKH23]. However, there's a notable difference in the encoding methods: while Fujieda et al. employ a grid-based method for encoding directions, our method utilizes Spherical Harmonics for this purpose. We observe that the *Neural Visibility Cache* (NVC) achieves significantly better reconstruction results compared to the NIRC. This improvement directly contributes to lower variance in the unbiased two-level estimator, enhancing the overall performance for environment map lighting.

To sum up, our contributions are:

- the introduction of the *Neural Incident Radiance Cache* (NIRC)
- a *Multi-Level Monte Carlo* (MLMC) estimator to remove bias
- new evidences suggesting that the MLMC framework in combination with the NIRC can achieve superior variance reduction in less training time over CV
- a comprehensive analysis comparing NRC and NIRC
- a set of performance optimizations enabled by the NIRC concept
- a specialized solution for environment map lighting.

2. Background

Light transport simulation The rendering equation [Kaj86] describes how light interacts with a scene. The radiance $L_o(x,\omega_o)$ leaving a point x in direction ω_o is:

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^+} L_i(x,\omega_i) f_r(x,\omega_i,\omega_o)\cos\theta_i d\omega_i, \quad (1)$$

where $L_e(x,\omega_o)$ is the radiance emitted at x in direction ω_o. The reflected light is computed by integrating over all incident light directions H^+. $L_i(x,\omega_i)$ is the incident light, $f_r(x,\omega_i,\omega_o)$ is the bidirectional reflectance distribution function, and $\cos\theta_i$ is the foreshortening. A typical Monte Carlo estimator for $L_o(x,\omega_o)$ is:

$$\hat{L}_o(x,\omega_o) \approx L_e(x,\omega_o) + \frac{1}{N}\sum_{i=1}^{N} \frac{L_i(x,\omega_i) f_r(x,\omega_i,\omega_o)\cos\theta_i}{p(\omega_i)}, \quad (2)$$

where N is the number of sampled light paths, and $p(\omega_i)$ is the probability density function used to sample incident directions ω_i.

Multi-Level Monte Carlo *Multi-Level Monte Carlo* (MLMC) methods [Gil08; Gil15] have been applied in various fields, but have not yet found widespread application in rendering. In this paper, we introduce a two-level Monte Carlo scheme (a subset of MLMC) to turn the integration based on our NIRCs into an unbiased estimator. We begin with the standard Monte Carlo estimator for an integral F over a domain $\mathcal{D}$:

$$F \approx \frac{1}{N}\sum_{i}^{N} \frac{f(X_i)}{p(X_i)}, \quad (3)$$

where $\{X_i\}$ are samples drawn from a probability distribution $p(x)$.

Our two-level MC consists of 1) a cache-based estimator F_c with an approximating function $f_c(x,\mathbf{w})$ and optimizable parameters $\mathbf{w}$:

$$F_c \approx \frac{1}{N_c}\sum_{i}^{N_c} \frac{f_c(X_i,\mathbf{w})}{p_c(X_i)}, \quad (4)$$

and 2) a residual error estimator F_r:

$$F_r \approx \frac{1}{N_r}\sum_{i}^{N_r} \frac{f(Y_i) - f_c(Y_i,\mathbf{w})}{p_r(Y_i)}. \quad (5)$$

We obtain the full two-level estimator F_{tl} as

$$F_{tl} \approx F_c + F_r, \quad (6)$$

where $X_i \sim p_c(X_i)$ and $Y_i \sim p_r(Y_i)$. Note that we can choose the number of samples for each estimator, N_c and N_r, individually. If the optimizable function f_c approximates the original function f well and is cheap to evaluate we can allocate more computation to the estimator F_c using more samples N_c per frame. The basis for f_c is the Neural Incident Radiance Cache (NIRC), which will be introduced and defined in Section 4.

Control Variates (CV) A control variate $g(x)$, with a known expected value $E[g(x)]$, can be used to estimate the integral of a function $f(x)$ with:

$$E[f(x)] \approx E[g(x)] + \frac{1}{N}\sum_{i=1}^{N} \frac{f(X_i) - g(X_i)}{p(X_i)}, \quad (7)$$

where X_i are points sampled according to a probability distribution function p. Control Variates show a certain similarity to Multi-Level Monte Carlo, however, they differ in that the expected value $E[g(x)]$ is computed analytically. This is also an inherent limitation: CVs require a mathematical framework for analytical integration over the domain. In our case, the expected value is derived from the cache-based integral in a MLMC framework.

3. Related Work

Radiance Caching Since the seminal work of Ward et al. [WRC88], numerous advances have been made in caching techniques. Important developments include the widely employed irradiance volumes [GSHG98] and the introduction of radiance caching [KGPB05] using spherical harmonics for directional domain representation. These methods have seen significant enhancements for both offline [DBN17; MJJG18; ZBN19] and real-time rendering [MGNM19], and new approaches to processing and storing lighting information [VPG14; SL17; RZD14; BFK18; Pan20; SNRS12]. The popularity of neural networks leads to the Neural Radiance Cache (NRC) [MRNK21] which offers a method for dynamic scenes, learning during rendering and leveraging low-cost computation over memory access by utilizing fully-fused neural networks. A combination of NRC with multiresolution hash encodings significantly accelerates training convergence while improving the quality of radiance signal reconstruction [MESK22]. Moreover, pre-trained neural networks have been shown to support dynamic parameters such as mesh positions or light parameters [RBRD22; DPD22] and still predict a reasonable approximation of global illumination. In contrast to the aforementioned works, we focus on efficient caching of incident radiance, a step that significantly improves rendering performance and lets us use the cache for an efficient unbiased Monte Carlo estimator for real-time rendering.

Neural Methods Neural Methods comprise a variety of techniques: Xie et al. [XTS*22] provide a detailed analysis of such techniques with an emphasis on *Neural Radiance Fields* (NeRFs) [MST*20]. Neural Radiosity [HCZ21] has parallels with traditional radiosity techniques but computes a solution to the rendering equation by applying neural networks to minimize the residuals. Additionally, deep neural networks can be used for guiding path tracing by generating scattering directions [MMR*19; ZXS*21; VKJ19] and by this enhancing the efficiency of Monte Carlo integration in light transport simulation.

Neural Control Variates (NCVs) [MRKN20] NCVs use Normalizing Flows to model Control Variates. These models are primarily chosen for their capability to ensure that the final integral of the function equals to 1 multiplied by a prediction of another neural network. Despite their theoretical appeal, experimental results on the performance of Normalizing Flow models [MRKN20; MMR*19; ZRW*23] have demonstrated notable computational demands for both training and inference. Moreover, NCVs require to maintain a second neural network which predicts the integral coefficient. These resource requirements often exceed what is practical for real-time or interactive rendering.

One motivation for our work was the potential advantage of MLMC over (N)CV in reducing the variance of the residual error. This requires a computationally efficient mathematical basis for numerically estimating the integral, which we achieve with NIRCs.

4. Neural Incident Radiance Cache

The computation of $L_i(x,\omega_i)$ is the most resource-intensive part of Equation (1). This is due to the need to trace rays, find intersection points, fetch surface material parameters and evaluate Equation (1) for a set of positions and directions, which often results in noisy estimations. In this work, we address this problem by firstly splitting $L_i(x,\omega_i)$ term into indirect lighting part $L_{ind}(x,\omega_i)$ and direct $L_{nee}(x,\omega_i)$:

$$L_i(x,\omega_i) = L_{ind}(x,\omega_i) + L_{nee}(x,\omega_i) \tag{8}$$

since we will use MIS combination with next event estimation for $L_{nee}(x,\omega_i)$. Secondly, we propose a cache that approximates $L_{ind}(x,\omega_i)$ via a neural network while computing the other terms normally:

$$L_{ind} \approx f_c(x,\omega_i,\omega_o,\phi,\mathbf{w}) = n_i(x,\omega_i,\phi,\mathbf{w}) f_r(x,\omega_i,\omega_o)\cos\theta_i. \tag{9}$$

Here $n_i(x,\omega_i,\phi,\mathbf{w})$ is a neural network with optimizable parameters (weights) $\mathbf{w}$, conditioned on a surface position x, incident vector ω_i, and a set of additional features ϕ provided for facilitating the search of correlation between target values and input parameters. This function n_i serves as the *Neural Incident Radiance Cache* (NIRC), approximating incoming lighting from all (hemi)-spherical directions onto a shading point. The overall architecture of the NIRC is illustrated in Figure 3 and explained in the following sections.

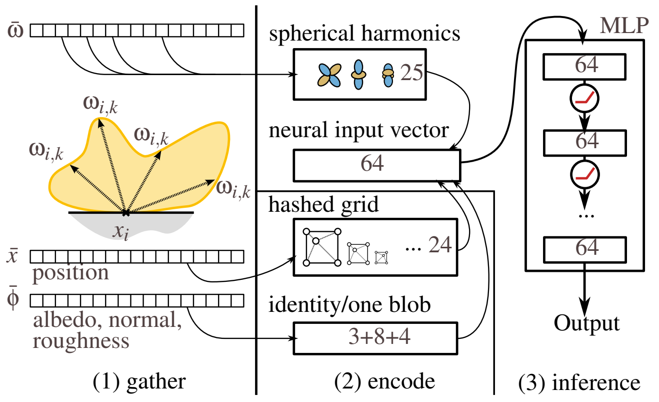


Figure 3: *The stages of our neural inference pipeline. Initially, requests in the path tracing pass are gathered (1), utilizing multiple buffers to store per-surface parameters along with the incident direction vectors ω_i. Next, the per-surface parameters are encoded (2) and aggregated. Finally, the spherical harmonics coefficients for the requested ω_i are calculated and appended in shared memory as input for the multi-layer perceptron inference (3). Our optimized implementation never stores the neural input and output vectors in global memory. Variables with a bar ($\bar{x}$, $\bar{\omega}$, etc.) represent arrays of the corresponding parameters.*

Using Equation (9), we are able to formulate an unbiased Multi-Level Monte Carlo estimator for computing the rendering equation:

$$\hat{L}_o(x,\omega_o) \approx L_e(x,\omega_o) + \hat{L}_{nee}(x,\omega_o) + \hat{L}_c(x,\omega_o) + \hat{L}_r(x,\omega_o) \tag{10}$$

$$\hat{L}_r(x,\omega_o) \approx \frac{1}{N_r}\sum_{i=1}^{N_r} \frac{(L_{ind}(x,\omega_i) - n_i(x,\omega_i,\phi,\mathbf{w}))f_r(x,\omega_i,\omega_o)\cos\theta_i}{p(\omega_i)} \tag{11}$$

$$\hat{L}_c(x,\omega_o) \approx \frac{1}{N_c}\sum_{i=1}^{N_c} \frac{n_i(x,\omega_i,\phi,\mathbf{w})f_r(x,\omega_i,\omega_o)\cos\theta_i}{p(\omega_i)}\omega_o), \tag{12}$$

correspondingly a biased estimator with the omitted residual error term:

$$\hat{L}_{biased}(x,\omega_o) \approx L_e(x,\omega_o) + \hat{L}_{nee}(x,\omega_o) + \hat{L}_c(x,\omega_o). \tag{13}$$

For more details on how the biased and unbiased estimators are applied in combination with the NIRC, please refer to Section 6.2.

4.1. Radiance Cache Optimization

To effectively optimize the trainable parameters $\mathbf{w}$, we must derive a suitable and practical loss function. Recall that Multi-Level Monte Carlo (MLMC) methods aim to reduce the overall variance of the estimator. This reduction is achievable when the variance of the residual estimator is minimized. The residual estimator F_r Equation (5), which can be defined as $F_r = \mathbb{E}\left[\frac{f(X)-f_c(X,\mathbf{w})}{p_r(X)}\right]$, represents the expectation of the normalized difference between the function f and its controlled approximation f_c. The variance of F_r is given by

$$\mathbb{V}(\langle F_r\rangle) = \int_{\mathcal{D}} \left(\frac{f(x)-f_c(x,\mathbf{w})}{p_r(x)} - F_r\right)^2 p_r(x)\,dx. \tag{14}$$

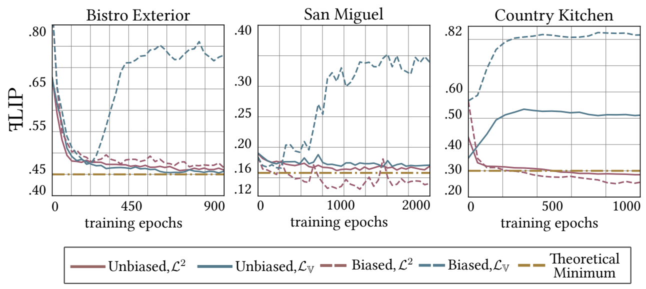


Figure 4: *We explore the implications of employing $\mathcal{L}^2$ and variance-based $\mathcal{L}_{\mathbb{V}}$ loss functions for the neural cache optimization with our two-level MC estimator and parameters $N_c = 4$ and $N_r = 1$ when the scene is rendered using the biased and unbiased estimators. We compute the ꟻLIP error between our renders, utilizing Neural Incident Radiance Cache, and reference renders every training epoch. The results demonstrate a detrimental effect on the quality of renders of the biased estimator when the $\mathcal{L}_{\mathbb{V}}$ loss function is applied. This is because the variance-based loss does not necessarily minimize the residual F_r. The data also shows divergence during training, likely due to issues when estimating F_r accurately, which is required for evaluating $\mathcal{L}_{\mathbb{V}}$. Moreover, we have to estimate F_r with multiple samples which makes the variance-based loss impractical for training on-the-fly.*

We can derive a numerical one-sample estimator for this variance of $\langle F_r \rangle$ which is then utilized as a loss function:

$$\langle \mathbb{V}(\langle F_r \rangle) \rangle \approx \left(\frac{f(x) - f_c(x, \mathbf{w})}{p_r(x)} - F_r \right)^2 \frac{p_r(x)}{q_r(x)}. \tag{15}$$

Here, $q_r(x)$ denotes the probability density function of the sampling points used for estimating the variance. We employ the same sampling methods for estimating the variance in Equation (15) as well as for estimating the residual error in Equation (5), leading to $q_r = p_r$. This results in the final variance-based loss function:

$$\langle \mathcal{L}_{\mathbb{V}}(f, f_c, \mathbf{w}) \rangle \approx \left(\frac{f(x) - f_c(x, \mathbf{w})}{p_r(x)} - F_r \right)^2. \tag{16}$$

We observe that we essentially obtain a squared distance between the cache and ground truth function, offset by the residual error's expectation. This suggests the optimization problem has potentially many solutions for any possible shift value.

Empirical experiments shown in Figure 4 highlight the adverse impact of integrating the residual error's expectation into the optimization point, justifying our reliance on the basic squared difference loss function:

$$\langle \mathcal{L}^2(f, f_c, \mathbf{w}) \rangle = \frac{(f(x) - f_c(x, \mathbf{w}))^2}{p_r(x)} \tag{17}$$

Considering the high dynamic range of the data we divide the metric by the squared $f(x)$ to ensure higher gradient weights for dark regions of a rendering scene as in [LMH*18]:

$$\langle \mathcal{L}^2_{rel}(f, f_c, \mathbf{w}) \rangle = \frac{(f(x) - f_c(x, \mathbf{w}))^2}{p_r(x)(sg(f_c(x, \mathbf{w})^2) + \epsilon)} \tag{18}$$

Where ϵ safeguards against division by zero and $sg(z)$ denotes that gradient computations for operation z are not propagated.

4.2. Path Termination Heuristics

The original NRC uses the *Spread Angle Heuristic* (SPH) for path termination, which almost always results in path termination after the first bounce, as shown in Figure 12. This suggests that we can leverage our cache instantly at the primary bounce without the need for this intermediate bounce since our cache predicts incident radiance directly.

The NRC path termination heuristic is defined as follows:

$$a(x_1 \cdots x_n) = \left(\prod_{i=2}^{n} \frac{\|x_{i-1} - x_i\|}{p(\omega_i | x_{i-1}, \omega) \cos \theta_i} \right)^2, \tag{19}$$

$$a_0 = \frac{\|x_0 - x_1\|^2}{4\pi \cos \theta_1}, \tag{20}$$

where $p(\omega_i)$ is the BSDF sampling PDF and θ_i is the angle between ω_i and the surface normal at x_i. Paths are terminated if:

$$a(x_1 \cdots x_n) > c \cdot a_0, \tag{21}$$

where c is a hyperparameter equal to 0.01.

This heuristic is not directly applicable in our setting since we want the ability to terminate on the first directly visible path vertex. We propose a new criterion inspired by the balance heuristic [VG95], which we call the *Balanced Termination Heuristic* (BTH). The BTH leverages the strengths of NIRC while acknowledging its limitations, particularly with glossy details. It essentially computes the MIS weight of BRDF sampling vs. a virtual diffuse sampling scheme with N_c samples. We calculate the continuation probability P_s as:

$$P_s = \frac{p(\omega_i)}{p(\omega_i) + \frac{N_c}{\pi}}, \tag{22}$$

where $p(\omega_i)$ is the pdf of the sampled scattering direction, and N_c is the number of neural samples. The path termination criteria are as follows:

$$\begin{aligned} i = 1: &\quad \text{stop if } \xi > P_s, \\ i > 1: &\quad \text{stop if } \xi > P_s \text{ or } a(x_1 \cdots x_i) > c \cdot a_0, \end{aligned} \tag{23}$$

where $\xi \in [0, 1)$ is a uniformly distributed random variable. If terminated, the current surface is shaded using N_c directions sampled according to the surface BRDF, incorporating direct lighting with Monte Carlo integration using Multiple Importance Sampling (MIS) and Next Event Estimation (NEE). Delta reflections automatically continue tracing without termination. For a fair comparison with NRC, we use its heuristic for all surfaces x_i where $i > 1$, ensuring that paths are not terminated later than they would be with NRC.

4.3. Cache Application

Our algorithm follows the megakernel path tracing approach. For each vertex x_j, we generate N_c^j incident vectors ω_i, where N_c^j denotes the number of samples of the NIRC at each vertex x_j, to estimate the cached radiance term L_c. This process requires collecting all per-surface parameters and incident vectors ω_i. These values are used for both cache estimation and for calculating the additional terms in Equation (12) for ω_i. To estimate the residual error Equation (11), we need a set of directions ω_i which are uncorrelated to the directions used to compute Equation (12).

We use independent sampling of ω_i according to the surface BSDF for both L_c and L_r. Alternative sampling strategies like path guiding for residual integral estimation have been explored in other studies [MRKN20]. To keep our study focused, we only employ basic importance sampling here. While we can show an improvement over baseline methods with this approach already, it would be an interesting future extension to perform direction sampling by incident radiance here (for instance by rejection sampling with the NIRC).

After the ray tracing pass, the collected parameters are encoded, aggregated, and utilized in the NIRC inference. The results from the inference are combined with the remaining terms in Equation (12) to compute the final rendered image.

4.4. Neural Network Architecture

Inspired by previous work we leverage the power of fully-fused executed multi-layer perceptions (MLP) [MRNK21] to represent a radiance field. MLP can efficiently approximate signals defined over a five-dimensional manifold conditioned on surface a position (3D) and an outgoing direction (2D), even with real-time performance on GPUs. However the expressive power of a tiny MLP is limited, and its convergence rate is adversely affected if input parameters are forwarded without applying any nonlinear transformation to their domain space. Therefore, we employ encodings inspired by the following neural rendering literature [MESK22; MRNK21; VHM*22]:

- The shading point position x is used for querying a multiresolution hash encoding with trainable features. The resolution is adjusted based on the complexity of the scenes, but in the majority of our experiments, we used 12 levels with 2 features per level.
- To enhance the NIRC's precision in the spherical domain, we propose estimating spherical harmonics coefficients for the input parameter ω_i [VHM*22]. Our experiments suggest that using 4 bands is a good compromise between increasing the neuron count and maintaining the network's directional representational quality.
- As proposed by [MRNK21], using the shading normal, albedo, and roughness as additional input parameters ϕ for NIRC, without altering their corresponding encodings.

4.5. Performance Optimization

A notable challenge in integrating high-quality encodings before executing an MLP is performance degradation. Trainable features,

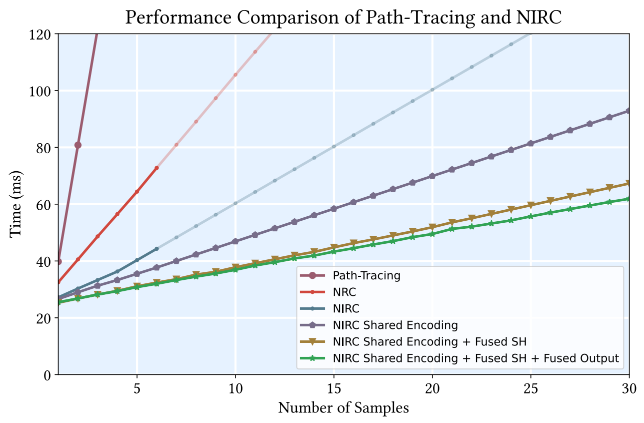


Figure 5: *Scalability and ablation study with the Bistro Exterior scene rendered on an RTX 3080 at 1080p. We show incident lighting computed using conventional path tracing and our NIRC; the latter has been tested with various optimization strategies. The results demonstrate improved scalability of the NIRC method, particularly when it is optimized by exploiting shared surface encodings (1.71⇑, indicating a relative performance improvement in average compute time per additional sample compared to the naive NIRC baseline), fused spherical harmonics coefficients computation (2.69⇑), and direct output forwarding from the shared memory using atomic variables (3.11⇑). For the NRC and the naive NIRC implementations, we show extrapolated performance timings after the 6th frame because of memory limitations (light blue and light red).*

used in hash encoding for shading point positions, require us to fetch numerous float variables from scattered memory locations. However, it is possible to amortize memory accesses across all requests for the same shading surface, given the way we use the cache (Equation (12)). This idea is extended to other shared input parameters ϕ and their corresponding encoding outputs. Furthermore, the computation of spherical harmonics coefficients for a relatively high number of bands can create bandwidth problems, moving 25 float variables into global memory per request. To alleviate this, we advocate for a fully-fused execution approach, not just for MLP inference, but also for the spherical harmonics encoding layer. Our algorithm calculates the necessary coefficients and immediately stores them in a section of shared memory reserved for future inference as input neurons. Additionally, we found that packing and compressing directional components maximizes system bandwidth.

Lastly, we suggest an optimization that avoids intermediate buffers used in the original tiny-cuda-nn framework during output: We write inference results into a final frame buffer directly from shared memory using atomic variables. This allows us to increase the number of neural samples per pixel within the time budget. We illustrate the final inference pipeline in Figure 3.

By integrating these natural optimizations, which emerge from our cache design choices and the manner in which we use it to estimate the Rendering Equation, we achieve up to 3.64× speedup compared to the original neural inference pipeline, depending on the sample count, as demonstrated in Figure 5. The performance cost of one neural sample can be as low as 2.69% of the corresponding path tracing simulation in our experiments. This significant performance improvement allows us to allocate more samples per pixel, reducing the variance of the stochastic estimator L_c.

For the training of our cache, we follow the algorithm of the original Neural Radiance Cache [MRNK21]. In our implementation, we use separate kernels to construct training and rendering paths. This decision is based on the following observation: using training paths derived directly from the main path tracing paths leads to performance issues in our implementation. Training requires us to trace paths of unbounded length to remain unbiased (by using Russian roulette). Since the main rendering paths are considerably shorter in our biased version, this results in thread divergence.

We found that executing a separate path tracing kernel for training paths not only simplifies the process but also slightly improves performance. While there will still be thread divergence due to differing path lengths, given that we only use about 2-3% the number of training paths as compared to the number of pixels, this additional pass does not incur substantial overhead. This method ensures efficient training of the cache while maintaining the integrity and performance of the main rendering process.

5. Neural Visibility Cache

In the previous section, we addressed the computation of incident illumination from other surfaces. While this approach works for the special case of lighting from an environment map, this type of illumination is important enough to warrant special treatment to improve efficiency. In particular, for environment map lighting we know that the incident radiance is the product of visibility V and environment illumination L_i: $V(x_i, x_{i+1})L_i(x_i, \omega_i)$. Since L_i is known from the scene definition we only have to find V. For that purpose, we propose a specialized *Neural Visibility Cache* (NVC) which is trained to only estimate the visibility term instead of the final radiance signal. V takes on values between zero and one (potentially including partial visibility due to transparency or transmittance in volumes), so we use a sigmoidal activation function on the output neurons of the NVC. Further details and empirical observations related to this approach are discussed in Section 6.2.

6. Results

6.1. Multi-level Monte Carlo vs. control variates

To assess the expressiveness of our proposed neural cache, we compare it to other analytical models, in particular *Spherical Harmonics* (SH) and mixtures of *von Mises-Fisher* lobes (vMF). These models serve as a basis in *control variates* (CV) approaches because they can be integrated analytically and have been used in previous work [Pan20]. To conduct a meaningful comparison, we allocate each pixel its own set of model parameters in a framebuffer at 720p, ensuring that every pixel holds the same amount of memory supporting the models. We chose 720p because we implemented this experiment in PyTorch [PGM*19] using Falcor 7 [KCK*22] and PyTorch Tiny CUDA NN bindings [Mül21]. This setup facilitates debugging and correct fitting of vMF and SH coefficients but limits the performance. We are planning to publish the source code for reproducing the experiments after the publication.

In our experiments, the vMF mixture model holds 11 lobes per pixel, requiring 7 coefficients per lobe. This model is similar to the Spherical Gaussian mixture model, which has been previously used as a basis for CV [Pan20]

We initialized the vMF lobes directions using a spherical Fibonacci lattice for optimal sphere coverage and optimized the lobes parameters using the stepwise-EM [VKŠ*14] [Pan20]. The batch size for the stepwise EM was set in a range from 15 to 40 samples, depending on the noise level of a scene.

We introduce *Screen Space NIRC* (SS NIRC), a version of NIRC that accepts one latent vector per pixel instead of the multiresolution hash encoding for a fair comparison with the vMFs and SHs models. The neural network additionally stores the network weights in a global buffer with less than 55kB and allocates 68.2 MB for the screen space latent features. The world-space NIRC requires only 20MB for its latent representation and 50kB for the weights respectively. We set both NIRC models to have 6 layers to assess their expressive power at full capacity.

We selected the number of vMF lobes to be 11, SH bands to be 5, and 74 latent vectors for SS NIRC to ensure a similar memory footprint of approximately 75 floats per pixel. We also limit the number of frames for optimization up to 4000 due to our target application goal of real-time rendering.

We estimate the relative variance of the residual error integral estimator $\mathbb{V}_{rel}(\langle F_r \rangle)$ for both control variates (vMFs, SH) and two-level Monte Carlo estimators (NIRC and SS NIRC) per each surface in the frame buffer and average them. We also provide visualizations of the computed integrand values by all four models for a set of selected surfaces. It is important to highlight that SHs and vMFs are trained to memorize not only the incident lighting signal as the NIRC and SS NIRC do but the whole integrand, including the BRDF term, so they can be used as control variates.

As seen in Figure 6, the neural models achieve lower variance for all scenes, preserving geometrical edges and reconstructing high-frequency illumination details. The standard NIRC with multiresolution hash encoding often reaches even better metrics by consuming less memory, as it effectively redistributes the limited latent features between surfaces by resolving hash collisions using gradient descent.

As shown earlier, analytical models like SH and vMF don't fully capture the complexity of the scenes and struggle to reconstruct the integrated illumination with reasonable accuracy. This motivated us to conduct further experiments with neural models, specifically implementing a model similar to *Neural Control Variates* (NCV) [MRKN20]. This model is based on a Normalizing Flow using the Piecewise-Quadratic Coupling Transform from Neural Importance Sampling (NIS) [MMR*19]. We utilized an open-source NIS implementation as the foundation for the warps with the invertible coupling transformations [DG21], and developed the rest of the components ourselves, as the original works did not release any public code [MMR*19][MRKN20]. Inspired by NCV, our NCV model utilizes one blob encoding for latent variables with 32 bins.

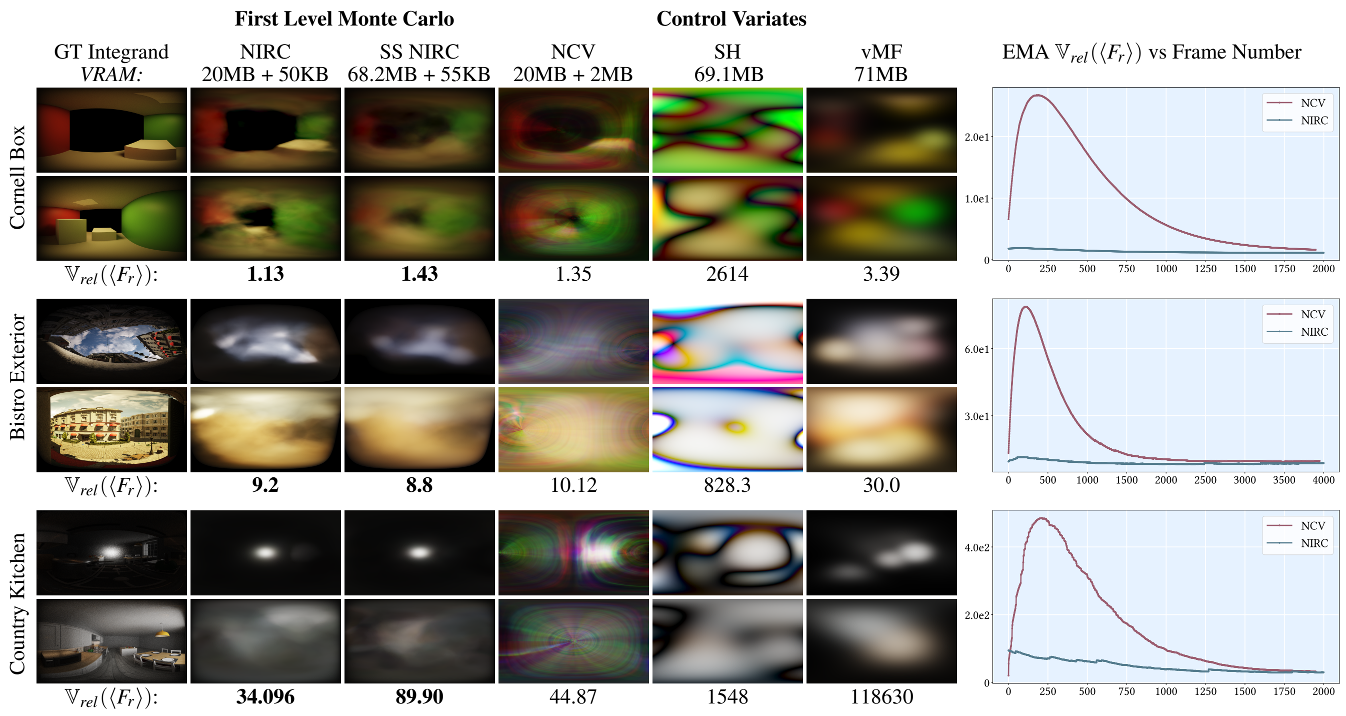


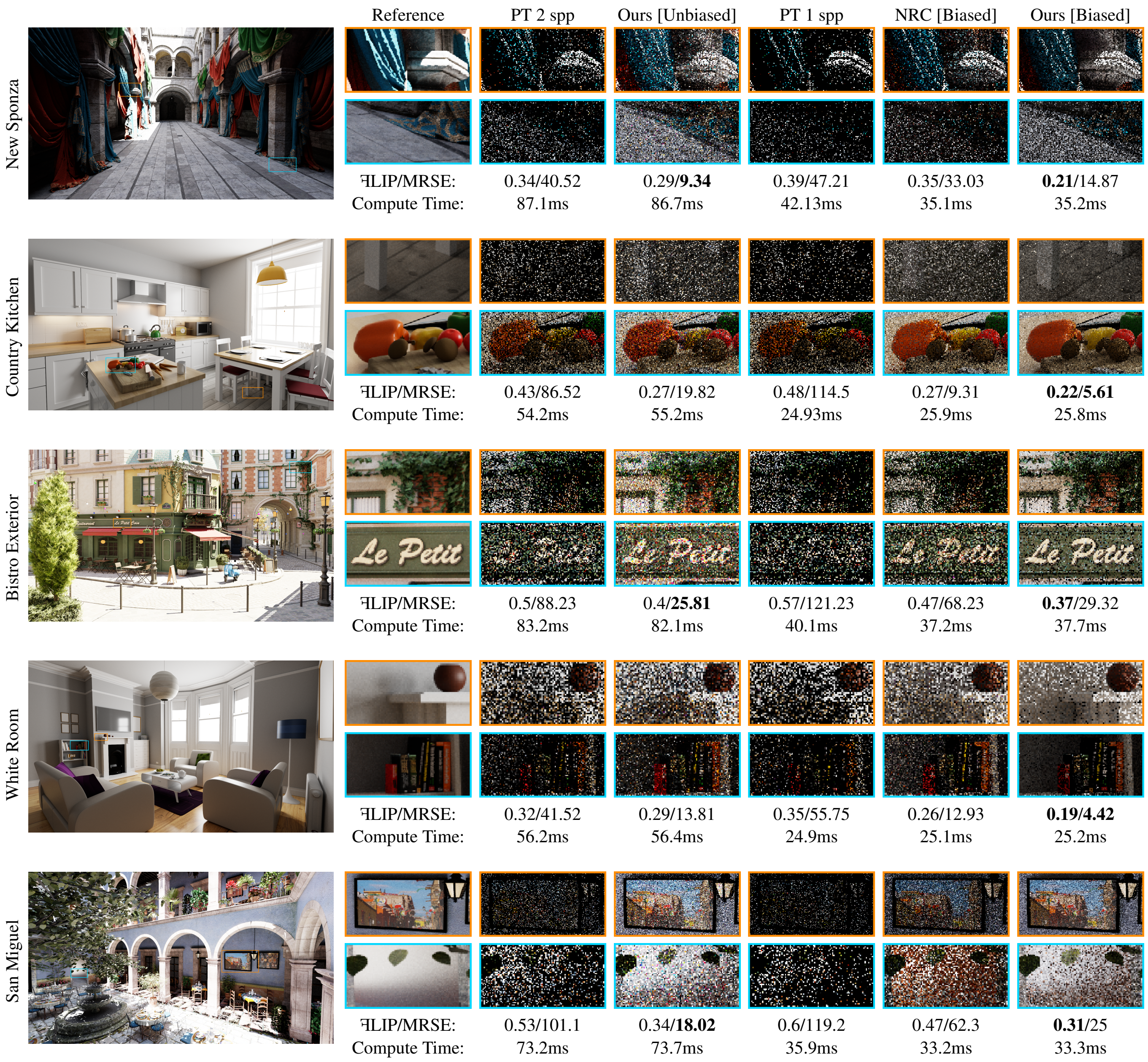
Figure 6: *Equal-memory footprint comparison with a static camera between SHs, vMF mixtures, Neural Control Variates [MRKN20] and two our neural caches: NIRC with world space multi-level hash encodings and screen space latent features (SS NIRC). This figure presents shading integrand values in selected surfaces with different materials in a set of scenes, displayed as tone-mapped latitude-longitude images. NCV, SHs and vMFs serve as control variates for analytical integral computations. NIRC focuses only on incident radiance on surfaces for use in MLMC estimators. SHs generally struggle with ringing and fail to capture high-frequency signals for specular surfaces. Although vMFs can theoretically adapt to any frequency, the stepwise EM algorithm can fail to optimize all lobes efficiently with limited noisy radiance samples. Neural Control Variates can evaluate integrals in closed form, but, as evidenced, the expressive power of NIRC demonstrates higher capacity. The average relative variance of the residual error estimator* $\mathbb{V}_{rel}(\langle F_r \rangle)$ *per each scene and method is presented under the images. The plots on the right show estimated* $\mathbb{V}_{rel}(\langle F_r \rangle)$ *using* Exponential Moving Average *(EMA) per training frame for the both neural models. As illustrated the NIRC requires significantly fewer training frames to achieve even lower* $\mathbb{V}_{rel}(\langle F_r \rangle)$ *than the NCV.*

The coupling transforms consist of 64 bins and 2 repetitions as in the original work. To ensure fair equal-memory comparison, we use the same encodings and the number of hash levels for the *optional features* (per-surface parameters) as for the NIRC. Our implementation of NCV optimizes the shape of the control variates using a 6-layer MLP with 64 neurons to predict the coupling matrices, similar to the NIRC. Directions are encoded using the cylindrical transformation. For the sake of simplicity, we instantiate an independent flow for each colour channel. While potentially increasing computation time, it demonstrates accuracy improvements as shown in [MRKN20].

Neural Control Variates also require CV integral optimization, so we employed the NRC [MRNK21] to predict the CV integral, using the same hash encoding layer for both models, ensuring a fair comparison. Additionally, we conducted experiments using a pre-estimated CV integral instead of employing a unified network for both tasks, but this approach did not yield any significant improvements in the overall performance metrics. Unlike NIRC, NCV needs a larger batch size (57,600 vs. 14,400) due to convergence issues, which is why we perform only 1 gradient descent step per frame for NCV compared to 4 for NIRC.

To compare the performance of NCV and NIRC, we track the relative variance per frame using an exponential moving average ($\alpha = 0.95$). The results are shown in Figure 6 (right). NCV comes close to the NRC metrics after a thousand frames, while the NIRC requires only a few hundred. Additionally, NIRC achieves in 1.1-1.32$\times$ lower relative variance for the residual error estimator as compared to the NCV.

Our experiments in Figure 6 indicate that the additional accuracy gained by numerically estimating the integral for MLMC can offset the benefits of an analytical solution for the first-level integral for CVs. Another advantage is that NIRC does not require the neural network to approximate the BSDF, as is necessary for the NCV, uses less compute, and does not require a CV integral prediction network. While it is not easy to assess the precise compute time for NCV inference, as it can strongly depend on the final optimized CUDA kernels, we estimate that our NCV model requires 18 MLP inferences compared to just one for NIRC alongside the coupling transformation.



Scene		PT 2 spp	Ours [Unbiased]	PT 1 spp	NRC [Biased]	Ours [Biased]
New Sponza	ꟻLIP/MRSE:	0.34/40.52	0.29/**9.34**	0.39/47.21	0.35/33.03	**0.21**/14.87
	Compute Time:	87.1ms	86.7ms	42.13ms	35.1ms	35.2ms
Country Kitchen	ꟻLIP/MRSE:	0.43/86.52	0.27/19.82	0.48/114.5	0.27/9.31	**0.22/5.61**
	Compute Time:	54.2ms	55.2ms	24.93ms	25.9ms	25.8ms
Bistro Exterior	ꟻLIP/MRSE:	0.5/88.23	0.4/**25.81**	0.57/121.23	0.47/68.23	**0.37**/29.32
	Compute Time:	83.2ms	82.1ms	40.1ms	37.2ms	37.7ms
White Room	ꟻLIP/MRSE:	0.32/41.52	0.29/13.81	0.35/55.75	0.26/12.93	**0.19/4.42**
	Compute Time:	56.2ms	56.4ms	24.9ms	25.1ms	25.2ms
San Miguel	ꟻLIP/MRSE:	0.53/101.1	0.34/**18.02**	0.6/119.2	0.47/62.3	**0.31**/25
	Compute Time:	73.2ms	73.7ms	35.9ms	33.2ms	33.3ms

Figure 7: *We show different scenes rendered at 1080p on an RTX 3080. The experiments highlight the benefits of our Neural Incident Radiance Cache (NIRC) in combination with the Multi-Level Monte Carlo (MLMC) approach. For an equal-time comparison, the number of neural samples per light path is manually adjusted per scene within the range of 15 to 25 for our unbiased estimator to match the run time of 2 spp path tracing. We compare our biased estimator, which is explicitly designed to bypass the computation of the residual error integral, with 1 spp path tracing and the Neural Radiance Cache (NRC) algorithm. One key difference is that NIRC demands fewer computational resources and leads to shorter path length compared to NRC. This allows for 3 to 7 neural samples compared to the single sample of NRC. Our results indicate a decrease in the Mean Relative Squared Error (MRSE) of path tracing by 1 to 2 orders of magnitude and a 1.5 to 2.5 times reduction in ꟻLIP relative to conventional path tracing. Moreover, even the efficiency comparison between NIRC and NRC based biased estimators leads to a 2× to 3× reduction in MRSE and a 1.2× to 1.5× decrease in ꟻLIP in favor of our method.*

6.2. Evaluation

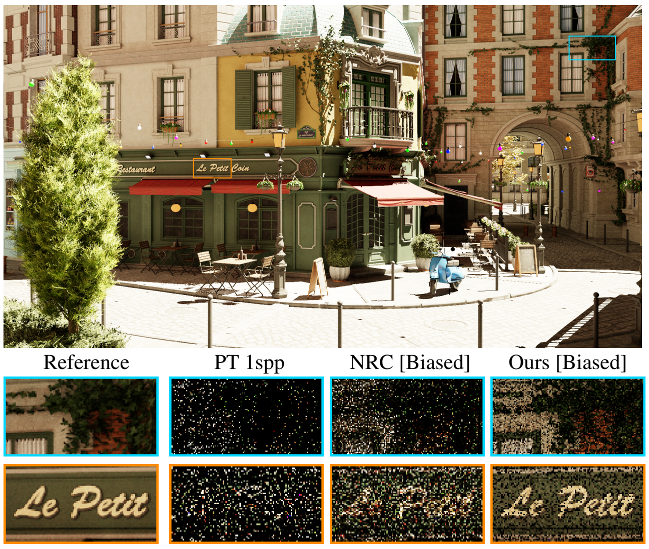
We implemented all components of our NIRC and the required rendering infrastructure for it using CUDA, based on an already released framework for training and inferencing fully-fused neural networks [MRNK21], and Direct3D 12 with the hardware accelerated ray tracing leveraging the Falcor 4.4 engine [KCK*22].

In order to achieve reasonable performance without a significant cache quality sacrifice we suggest using neural networks with 4 hidden layers and 64-neuron width. For the training, we follow the same strategy as was suggested for NRC [MRNK21] but without using the self-training strategy for our NIRC, because we did not observe any benefits from it. Also, we perform 4 gradient descent steps per frame using the Adam optimizer [KB17] with a learning rate equal to 0.01. ReLU activation is used as a nonlinear block for the MLP. For the NRC we use just one cache query at a final surface x_t for each light transport path (render sample) in all experiments. The cache-based Monte Carlo estimator Equation (12) is applied only in combination with the NIRC (see Figure 2).

The main experiments were conducted on an RTX 3080 GPU and an i7-12700k CPU with 32GB of RAM, rendering at 1920×1080 resolution. For the sake of fair judgments, we performed equal-performance comparisons involving computations of Mean Relative Squared Error (MRSE) and the perceptual ꟻLIP-metric [ANA*20]. The rendering is performed by a unidirectional path tracing algorithm combined with Next-Event Estimation (NEE) and Multiple Importance Sampling (MIS) [VG95] for direct lighting estimation. Lights are sampled by a light BVH [MPC19]. The path-tracer uses Russian roulette with a termination probability of 0.1.

Biased and unbiased variants We examine two possible ways of using NIRCs to accelerate rendering: We achieve an unbiased estimator by combining NIRCs and MLMC. For this, we estimate the outgoing radiance for the first three vertices of a path using a predefined number of neural samples (manually adjusted per scene), as well as full path tracing samples to estimate L_r. We obtain a biased estimator when instead using the Spread Angle Heuristic criteria for the NRC and the Balanced Termination Heuristic for the NIRC. Here we compute the shading integral using only the neural approximation of the incident lighting for truncated paths instead of performing path tracing; this results in better performance, but introduces bias. The only difference to the NRC biased estimator is that we can afford to compute the shading integral with more than 1 cache sample, as the NIRC allows querying additional neural samples without requiring any additional rays to be cast. As can be seen in Figure 7, our unbiased estimator reduces the MRSE by a factor of 3 to 5× and ꟻLIP by 10-38% compared to path tracing. Our biased estimator reduces MRSE by 3.17 to 20.40×, and achieves a 35-54% reduction for the FLIP metric. Compared to the NRC estimator, our biased algorithm also demonstrates significant variance reduction in the same computation time: ꟻLIP decreases up to 34% and MRSE is reduced by 1.66 up to 2.92×.

Environment map lighting Our cache includes special support for environment map lighting. This approach improves our method and thus Figure 7 does not compare the pure approximation power of the NIRC and NRC, since NRC does not decrease the variance caused by direct environmental lighting at the primarily visible surfaces. Figure 8 shows a balanced comparison with the NRC. Even when the influence of sky lighting is eliminated, our biased estimator exhibits outperforms the NRC.



	PT 1spp	NRC [Biased]	Ours [Biased]
ꟻLIP/MRSE:	0.55/112.4	0.45/82.5	**0.37/68.3**
Compute Time:	39.4ms	37.0ms	37.4ms

Figure 8: *We present a comparative analysis of the performance between our Neural Incident Radiance Cache (NIRC) and the standard Neural Radiance Cache (NRC) when rendering the Bistro scene with environment lighting disabled on a RTX 3080 and path tracing (PT). It ensures a fair comparison since the standard NRC does not mitigate the variance caused by sky illumination estimation. The performance of both techniques is evaluated based on two metrics: Mean Relative Squared Error (MRSE) and ꟻLIP. It is evident from the results that despite its inherent bias, our NIRC estimator consistently outperforms the NRC in the provided scenario.*

Distribution of neural samples in a light path Our approach can be used for estimating the outgoing radiance at each vertex along a light path. In our experiments with the unbiased estimator, we limit the NIRC usage to the first three non-specular vertices of a path for the sake of simplicity, although it is obvious that a light path's variance may be affected by other vertices. This decision is also influenced by the need to predefine the number of vertices that can sample NIRC, which is critical for managing corresponding memory allocations discussed in the following paragraph. Additionally, our experiments do not show substantial improvements when increasing the number of vertices beyond three. Figure 9 shows experiments on the distribution of the neural samples within a light path. Ideally, the amount of variance introduced at each vertex should be taken into account to determine the number of neural samples; we believe this is an interesting aspect for future work.

Memory consumption The memory consumption for NIRC inference is mainly determined by the number of surfaces where the NIRC is queried and the number of requests per surface. Our implementation includes several buffers: the *surface parameter buffer*,

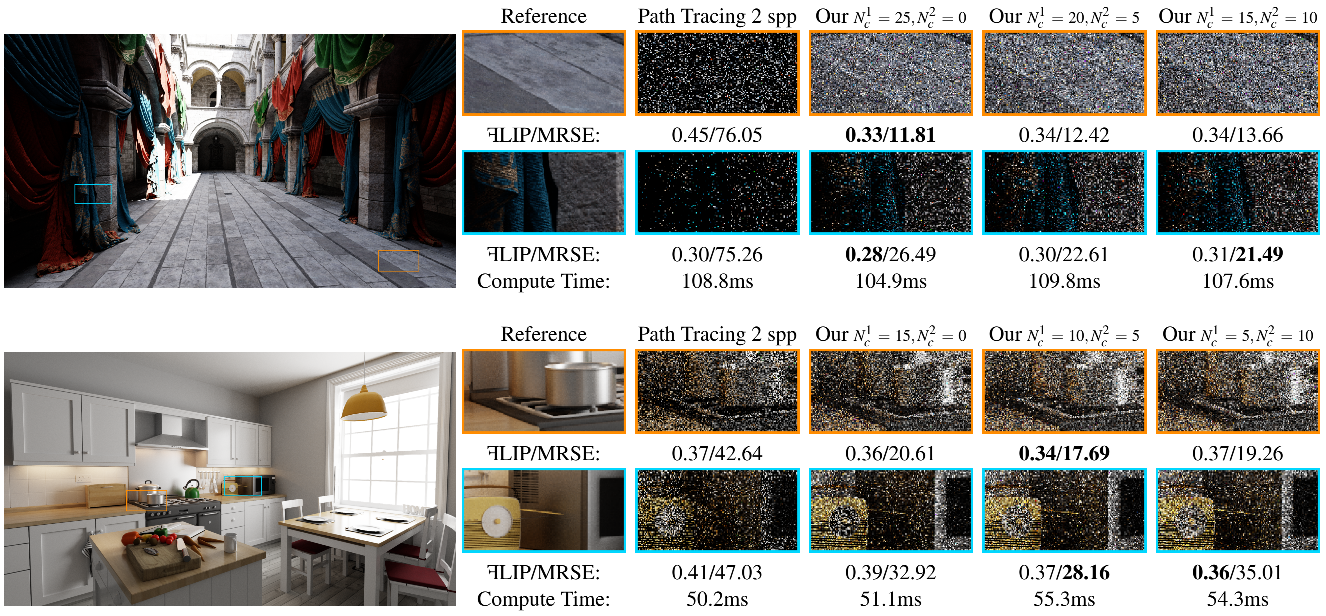


Figure 9: *We explore the effect of redistribution of the number of neural samples between the first and second vertex of a light path, denoted as N_c^1 and N_c^2, respectively. The experiments show the Sponza and Country Kitchen scenes rendered with up to 5 vertices in a light path without Russian roulette in order to match the render time. We observe that the optimal allocation of neural samples between the first and second vertex is highly scene-dependent, influenced by, for example, material properties and illumination.*

Scene	Render Paths	Training Paths	Model Training	Hash Encoding	Model Inference	Neural Samples
Bistro Exterior	16.5ms	1.2ms	5.27ms	2.53ms	6.29ms	7
Sponza	19.5ms	1.0ms	5.02ms	2.80ms	5.09ms	5
White Room	10.7ms	0.52ms	5.43ms	3.14ms	3.96ms	3
Country Kitchen	10.6ms	0.50ms	5.38ms	3.20ms	4.00ms	3
San Miguel	12.2ms	0.70ms	5.17ms	2.95ms	7.54ms	7

Table 1: *Average compute time breakdown of the main rendering stages and neural kernels for different scenes rendered at 1080p on a RTX 3080 with the biased estimator and varying number neural samples per pixel for the NIRC, corresponding to the biased renders in Figure 7. Model inference includes both neural network inference and the costs of all input encodings, excluding the hash-grid encoding tied to surface positions. The chosen number of neural samples per pixel is also shown.*

which requires 9 floats (36 bytes per surface), the *encoded parameter buffer*, utilizing 39 half floats or 78 bytes per surface and an additional 4 bytes for each direction, where a neural sample is requested, are stored in the *directional request buffer*. The unit direction vector is mapped to an octahedron, stored as 32 bits as in previous works [CDE*14; MSS*10]. In our experiments, the NIRC operates on up to 3 non-specular vertices per light path, with a total of 25 neural samples. This results in memory usage of 223.9 MB for the surface parameter buffer, 448.2 MB for the encoded parameter buffer, and 207 MB for the directional request buffer for 1080p resolution. We discuss potential memory optimization strategies in Section 7.

Performance Table 1 presents the compute time breakdown for the main rendering stages and neural kernels with the biased estimator and the NIRC. A significant portion of the time, 42.7%-51.9%, is spent on constructing rendering paths, where the shader traces the scene and fills in structured buffers with neural cache requests for processing by the CUDA backend. In contrast, training path construction accounts for only 2.1%-3.8% of the total time. Model training, which contributes 18.1%-22.9% of the time, is dominated by 4 gradient descent steps and can be easily adjusted for less dynamic scenes.

Dynamic scenes Our method demonstrates good performance even in non-static scenarios such as the Bistro Exterior scene with dynamic objects and a moving camera (Figure 10). We compare our real-time rendering using online trainable NIRCs in combination with the MLMC estimator to ground truth images computed using an offline renderer with 1000 samples per pixel (spp) for each frame. As visible in the plots, our method starts to outperform the standard Monte Carlo approach with the same compute budget after approximately 20 frames.

Note that our model is trained in an online fashion and is defined in world space. This means that with each new frame, the model can improve the quality of subsequent renders and show good generalization rather than an overfit to generated paths from

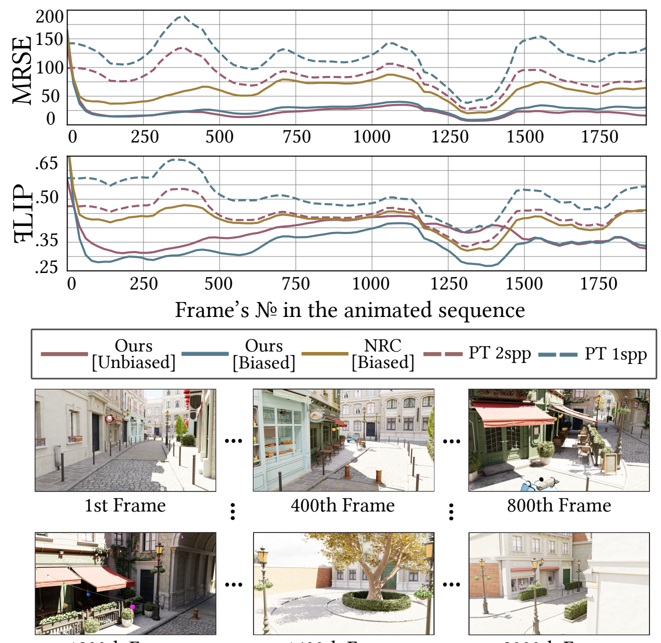


Figure 10: *To illustrate the dynamic adaptation of the* Neural Incident Radiance Cache *(NIRC) in a changing environment, we conducted experiments using the animated Bistro Exterior scene with moving objects. The figure demonstrates a quick reduction in both Mean Related Squared Error (MSE) and ꟻLIP metrics, reaching levels better than the conventional Monte Carlo estimator within a span of 10-20 frames. The compute time for path tracing (PT) with one sample per pixel (spp) and the biased estimators based on NIRC and NRC, as well as PT with two spp and our unbiased estimator, are comparable, respectively.*

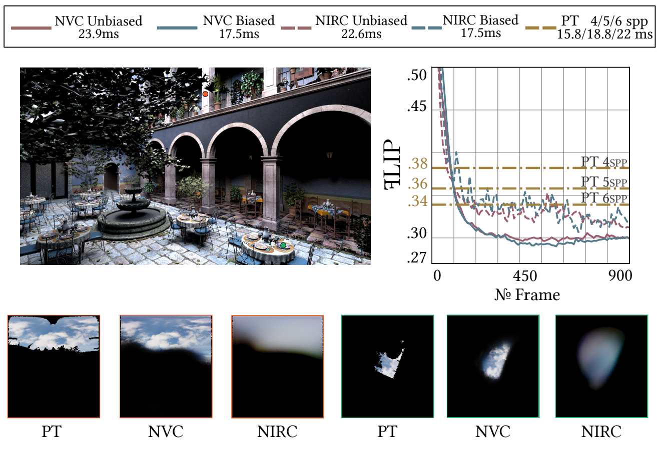


Figure 11: *We assess the effectiveness of the Neural Incident Radiance Cache (NIRC) and Neural Visibility Cache (NVC) to decrease the variance of the direct illumination estimator utilizing our MLMC estimator with $N_c = 7$ in the San Miguel scene. Despite using more memory bandwidth, the NVC exhibits better reconstruction quality and training stability than the NIRC. Since the baseline path tracing estimator is very fast in this setting, we limited the MLPs to 2 layers for both NIRC and NVC to achieve comparable timings.*

previous frames. We encourage readers to also watch the accompanying videos in the supplementary materials.

Neural Visibility Cache (NVC) Our results demonstrate that inferencing the visibility term instead of the final radiance value (Figure 11) outperforms basic path tracing as well as the generic NIRC. The benefits are particularly apparent when comparing the reconstructed hemispherical maps of incoming radiance for shading points. Focusing on learning the visibility term only also significantly simplifies the training process, and as demonstrated by the corresponding plot, NVCs converge more rapidly and provide more stable training compared to NIRCs. However, it is important to note that the NVC requires higher bandwidth due to the required sampling of the environment map; this is also illustrated in Figure 11.

6.3. Cache Analysis

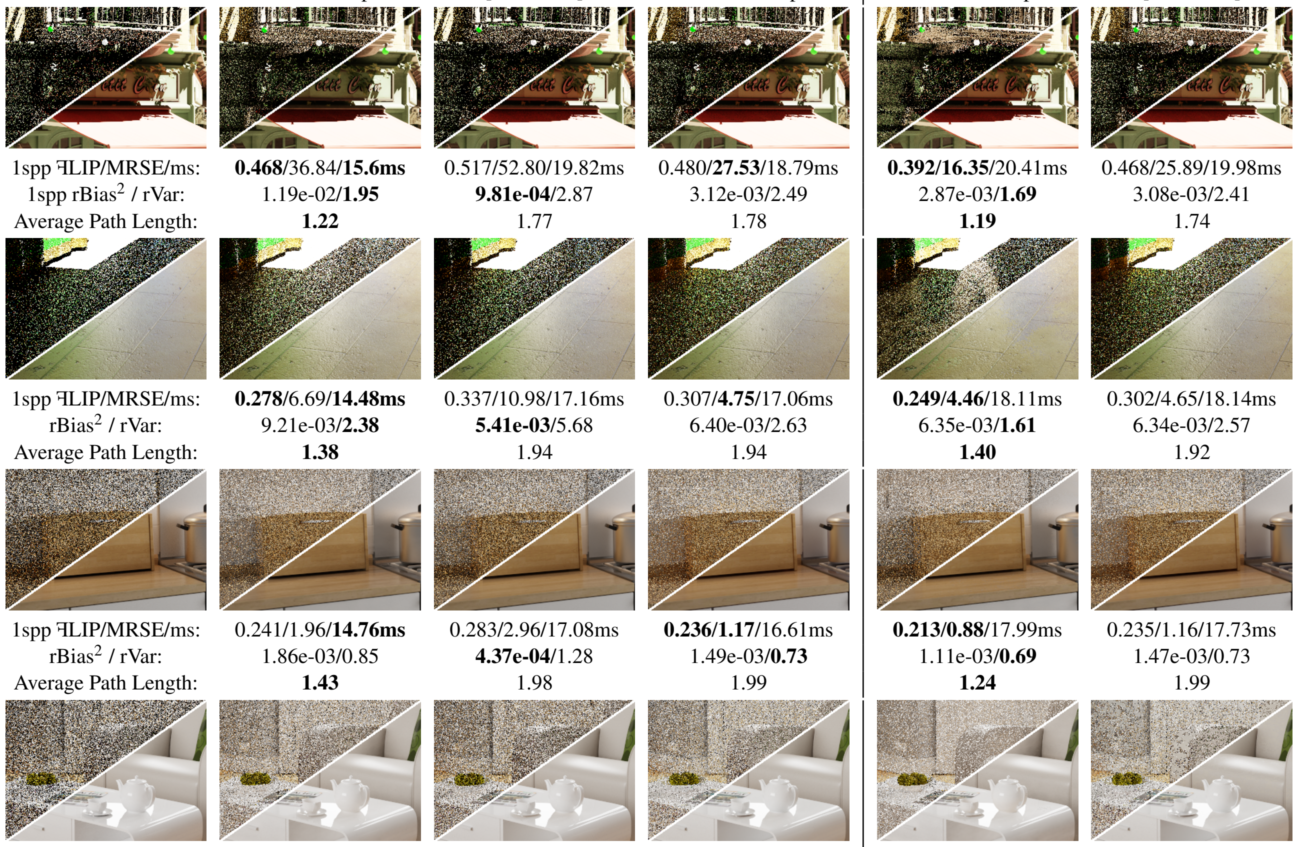
To assess the effectiveness of the NIRC in comparison to the NRC, we investigated their influence on bias, variance, path length, and render time.

While the NIRC allows us to achieve lower image errors within a limited sample budget by invoking it multiple times and saving bounces (Figure 5), it remains to be shown that this cache can approximate the incident radiance precisely enough to achieve converged renders. Moreover, our new heuristic may negatively impact the final render by causing most paths to terminate at the first vertex. This saves performance but can introduce bias. Therefore, we aim to answer several questions:

- How do different caches (the NIRC vs. the NRC) influence the bias and variance of the final estimator?
- Which cache produces more accurate renders for real-time rendering within limited compute time?
- How does the decision to stop earlier or later impact the bias vs. variance trade-off and render time?

To investigate these questions, we conducted a comprehensive analysis on a set of scenes using an RTX 4080, 32GB RAM, and an i7-12700K. For the NIRC, we only considered the task of capturing indirect light, excluding NVC and direct lighting estimation from an environment map for a fair comparison. We trained the caches for 2000 frames with 4 epochs to ensure their convergence.

As shown in Figure 12, relying on the *Spread Angle Heuristic* (SPH) for NIRC results in a lower *Relative Squared Bias* (rBias^2) of the final estimator by up to 4.12× compared with the NRC. However, this approach increases noise due to the need to integrate over the cache and estimate direct lighting, sometimes resulting in higher image-based error within a limited sample budget with just one sample per pixel. Conversely, our *Balanced Termination Heuristic* (BTH) demonstrates less noisy results for real-time rendering purposes and reduces the number of light bounces required for estimating indirect lighting by up to 3.54×, though this comes at the cost of increased bias in our estimator. Our analysis



		Heuristics Based Comparison			Approx. Equal Bias Comparison	
Scene	Reference	NIRC Ours BTH 1 neural sample	NIRC Ours + SPH [MRNK21]	NRC SPH 1 neural sample	NIRC Adaptive 1-5 neural samples	NRC Adaptive [MRNK21]
Bistro Exterior	1spp ꟻLIP/MRSE/ms:	**0.468**/36.84/**15.6ms**	0.517/52.80/19.82ms	0.480/**27.53**/18.79ms	**0.392/16.35**/20.41ms	0.468/25.89/19.98ms
	1spp rBias^2 / rVar:	1.19e-02/**1.95**	**9.81e-04**/2.87	3.12e-03/2.49	2.87e-03/**1.69**	3.08e-03/2.41
	Average Path Length:	**1.22**	1.77	1.78	**1.19**	1.74
Glossy Sponza	1spp ꟻLIP/MRSE/ms:	**0.278**/6.69/**14.48ms**	0.337/10.98/17.16ms	0.307/**4.75**/17.06ms	**0.249/4.46**/18.11ms	0.302/4.65/18.14ms
	rBias^2 / rVar:	9.21e-03/**2.38**	**5.41e-03**/5.68	6.40e-03/2.63	6.35e-03/**1.61**	6.34e-03/2.57
	Average Path Length:	**1.38**	1.94	1.94	**1.40**	1.92
Country Kitchen	1spp ꟻLIP/MRSE/ms:	0.241/1.96/**14.76ms**	0.283/2.96/17.08ms	**0.236/1.17**/16.61ms	**0.213/0.88**/17.99ms	0.235/1.16/17.73ms
	rBias^2 / rVar:	1.86e-03/0.85	**4.37e-04**/1.28	1.49e-03/**0.73**	1.11e-03/**0.69**	1.47e-03/0.73
	Average Path Length:	**1.43**	1.98	1.99	**1.24**	1.99
The White Room	1spp ꟻLIP/MRSE/ms:	0.308/6.81/**14.45ms**	0.351/8.83/17.52ms	**0.293/4.11**/16.43ms	0.269/**0.54**/17.04ms	**0.201**/1.98/17.60ms
	1spp rBias^2 / rVar:	1.50e-03/0.67	**7.73e-04**/0.89	3.19e-03/**0.57**	1.54e-03/0.50	2.90e-03/**0.33**
	Average Path Length:	**1.40**	1.96	1.99	**1.05**	1.56

Figure 12: *Comparison of the* Neural Incident Radiance Cache *(NIRC) and the* Neural Radiance Cache *(NRC) using different heuristics and adaptive methods rendered on an RTX 4080 on a set of scenes at 1080p with biased estimators. The left part of the figure provides a heuristics-based comparison, showing our NIRC with the* Balanced Termination Heuristic *(BTH) and the* Spread Angle Heuristic *(SPH) against NRC SPH, each using 1 neural sample. The NIRC achieves the lowest bias in all scenes with SPH and even when the paths are terminated at the 1st visible surface in Bistro and The White Room. The right comparison provides a fair representation of the performance of the algorithms with closely similar bias based on the per-pixel adaptive path termination policy. The figure shows that our NIRC method reduces the* Mean Relative Squared Error *(MRSE) and ꟻLIP metrics in scenes with a high influence of indirect lighting on the final variance while terminating the paths earlier and using limited computational resources more efficiently than NRC.*

shows that while invoking the NIRC at the same location as the NRC can lead to superior bias reduction, it degrades other metrics because we need to stochastically estimate the outgoing radiance from there, incorporating indirect lighting based on random samples of the cache along with direct lighting estimation. To gain theoretical insight, we explore an adaptive strategy for applying NIRC and NRC caches during rendering to analyze overall performance and a range of characteristics.

The experiment is set up as follows: We estimate the relative bias per pixel and decide whether to query the cache at the first visible path vertex based on this. If the relative bias of the converged render terminating the path at the cache compared to the ground truth is less than ε, the cache is used. By iterating over a set of ε values, specifically $\{0.05, 0.1, 0.15, 0.20, 0.25, 0.30, 0.35, ..., 1.0\}$, we collected a comprehensive set of metrics: MRSE, ꟻLIP, compute time for one-sample renders, (rBias^2), relative variance (rVar), and the average index of a vertex at which the cache is invoked (this is either the first or the second path vertex). We calculated at least 5,000 samples per pixel up to 50,000 to achieve fully converged ground truth renders and error maps. This was done for both the NIRC and the NRC. We excluded NIRC usage at v_1 for delta scattering rays and the NRC if the roughness is less than 0.0625 due to extreme glossiness. The NRC authors suggest a similar heuristic to invoke

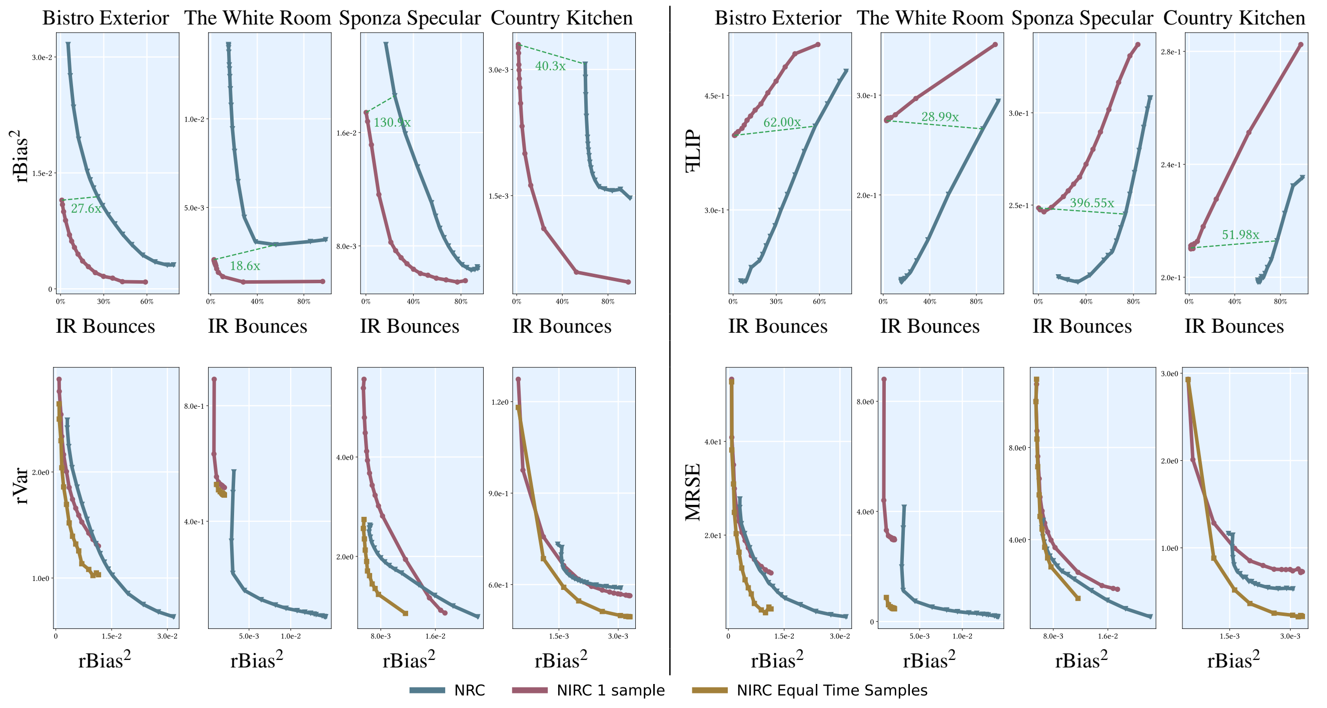


Figure 13: *For each scene, we analyze the performance of the* Neural Incident Radiance Cache *(NIRC) and the* Neural Radiance Cache *(NRC), highlighting the significant reduction in* Indirect Radiance *(IR) bounces and improvements in* Mean Relative Squared Error *(MRSE) and ꟻLIP based on the per-pixel adaptive path termination policy. The results show that in most scenes the NIRC curves (in red and brown) are consistently below the NRC curves (in red), indicating superior performance. Additionally, we include a comparison of the NIRC with equal compute time to NRC (brown) that maximizes the number of allocated neural samples per path for the NIRC, demonstrating the accuracy gains and variance reduction achieved by our algorithm. In particular, plots show a significant reduction in IR Bounces over the NRC for achieving closely similar bias and ꟻLIP, as highlighted by the green dashed lines in extreme cases with up to 396x difference. These improvements suggest that NIRC achieves lower error and variance at a similar computational cost to NRC.*

the cache only at the first *"non-specular vertex"* [MRNK21] as motivation for the SPH. For v_i where $i > 1$, if the error pass fails or the first interaction is "too glossy", we always follow the Spread Angle Heuristic for both cases. The algorithm uses the screen space error estimation approach for maximum precision, understanding that this approach limits the ability to guide the termination policy for the second and further bounces. However, it remains the most accurate method for the initial termination decision.

As shown in Figure 13, we introduced a new metric, *% Indirect Radiance* (IR) Bounces, representing the averaged index of a vertex at which the path was invoked minus one. Our results show that in extreme cases, when the render quality requirements are significantly relaxed, we can achieve up to 130.9× fewer IR bounces for estimating indirect illumination while maintaining a similar bias in the final estimator as for the NRC. Furthermore, we consistently achieved lower bias for any % IR bounces and lower variance in all cases where NRC can achieve the same bias. In the other cases, NRC is simply not able to provide a precise approximation high enough to have the same bias level as NIRC without delaying further the path termination decision. Moreover, it can be seen that a set of experiments leads to a significant saving in the number of IR bounces (see the dashed green lines) compared to the NRC depending on the scene when we consider equal-error comparisons based on ꟻLIP with just 1 spp. We may want to emphasize that these results are produced when NIRC leads to allocating IR bounces only to less than 0.2% of pixels while the NRC needs to trace further in more than 10% of pixels to achieve the same ꟻLIP metric or the rBias^2 of the final estimator and the rendering quality was traded off. A more practical set of cases, suitable for production applications with reasonable rendering quality, is shown in Figure 12 and discussed above.

Despite not exploiting the main architecture feature of the NIRC and allocating only 1 neural request per path for the NRC as well, we focused on more fundamental metrics regardless of the final compute time. We were also interested in applied cases for real-time applications. This is why we introduced the NIRC Equal Time Samples plot in Figure 13. We suggested maximizing the number of allocated neural samples for the NIRC per pixel where the cache is invoked at the v_1 until we match the compute time of the corresponding render with the closest rBias^2 based on the NRC (within a 1ms epsilon). By having more neural samples, we managed to achieve FLIP: 1.05-1.14× decrease, MRSE: 0.93-6.67× improvement, Variance: 1.11-1.82× improvement over the constant 1 neural sample. This improvement lets us achieve superior MRSE and rVar compared to the NRCs almost in all cases with equal bias. So, even considering the need to trace direct light rays and the associated costs, efficient adaptive usage combined with cheaper neural samples dominate over these costs.

This adaptive error-based algorithm requires a lot of precomputations and cannot be used in real-time applications. We discuss it to demonstrate potential theoretically; in reality, a classical hash table based data structure could be used to estimate the error gradually. Alternatively, very cheap Bayesian uncertainty estimation [DDLF24] does not require any additional infrastructure and could be theoretically used to estimate the bias.

7. Discussion and Future Work

Examining the quality of NIRC The decision to store incident radiance in our cache, as opposed to outgoing radiance in Neural Radiance Caching (NRC), is an inherent challenge for the model as it effectively has to learn the geometrical structure of a scene. As we discussed earlier this can lead to missed local reflection effects and impact performance with highly specular surfaces. When comparing converged renders using the *biased* estimators based on NRC and NIRC, the latter tends to produce renders of lower accuracy (the unbiased estimator does not share these difficulties). A venue for future research might be to enhance the quality of NIRC in the angular domain or to explore ways to combine NIRCs and NRCs to mitigate their limitations and achieve a better balance between performance and quality of biased renders.

MLMC Adaptivity MLMC removes the bias from the estimator but can increase variance. This is evident from Figure 14 where NIRC struggles to accurately represent incident light patterns for complex geometry like trees and bushes. We believe it is worthwhile to explore ways to improve MLMC's adaptivity. Müller et al. [MMR*19] have demonstrated the advantages of using a separate neural network to calculate the probability of relying on a trained neural guiding model instead of a regular sampling routine. Similar strategies might be used to restrict the use of MLMC to configurations where the cache quality is high.

Quality of caches Undesirable results may emerge, particularly in regions with high variance of radiance estimates, as the training of neural networks can become unstable due to noisy gradients. We believe that the use of techniques such as ReSTIR [BWP*20] might lead to better cache quality and reduce the occurrence of undesirable patterns in the render.

Dynamic adaptivity Although our algorithm typically performs well in dynamic scenes, the neural network might reach a local minimum and cease further adaptation. This can also be observed even when stochastic gradient descent [Rud16] is used without adaptive estimation of the gradient's momentum. We believe this deserves further work in the future.

More than two-levels Monte Carlo In our work, we have studied a two-level Monte Carlo method. The experiments in Figure 15 indicate that often 2-layer MLPs reproduce the original signals quite well. Therefore, it might be beneficial to use such MLPs as a first

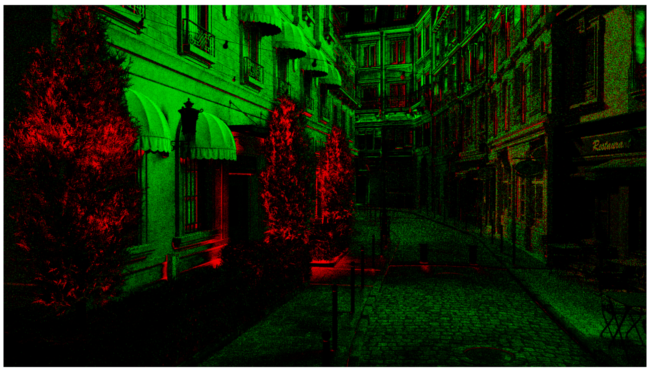


Figure 14: *This image shows a tone-mapped visualization of signed squared differences between the standard deviation of the original Monte Carlo and our multi-level Monte Carlo estimator for direct illumination only. Negative outcomes (red) indicate that our approach increases variance and positive outcomes (green) signify variance reduction. Our method works best in smooth regions and can fail near complex geometric occlusion such as in trees and bushes.*

level, and further MLPs, possibly with varying reconstruction qualities, in higher levels of MLMC.

Negative values The rendering equation does not yield negative values, ensuring that our NIRC model consistently predicts positive values, making it well-suited for our applications. However, when considering the residual error integral in our two-level Monte Carlo estimator, we encounter the possibility of negative values. Unlike other works [MRKN20; MMR*19], we do not employ path guiding, as it is considered orthogonal research. This absence of path guiding might result in slower convergence due to sign-based variance. While techniques like the positivisation method [OZ00] could potentially mitigate this issue by addressing the negative values, implementing such techniques would necessitate learning distributions, fitting Control Variates or path resampling. The latter would involve tracing multiple paths from a vertex where we invoke the cache, complicating the approach and degrading the performance of the whole algorithm. Therefore, we acknowledge this as an important area for future research.

Challenges in Optimization Our experiments showed that using L2 differences as a loss function does not consistently lead to the best outcomes. Taking into account tone mapping, or even perceptual metrics, might be beneficial, however, non-linear functions lead to biased gradients due to the interference of noisy radiance estimators. Previous work [NRN*23; CSN*23] proposes to decrease the variance of estimators, which in turn reduces the bias of the gradients. We believe that this is an interesting direction for research.

In Place Execution Our inference pipeline temporarily stores all neural network requests in buffers, a topic we discuss in detail in Section 6.2 concerning memory consumption. This approach not only requires substantial memory but also increases the consumption of memory bandwidth, which may lead to performance degradation. Keeping CUDA and Direct3D in sync within our implemen-

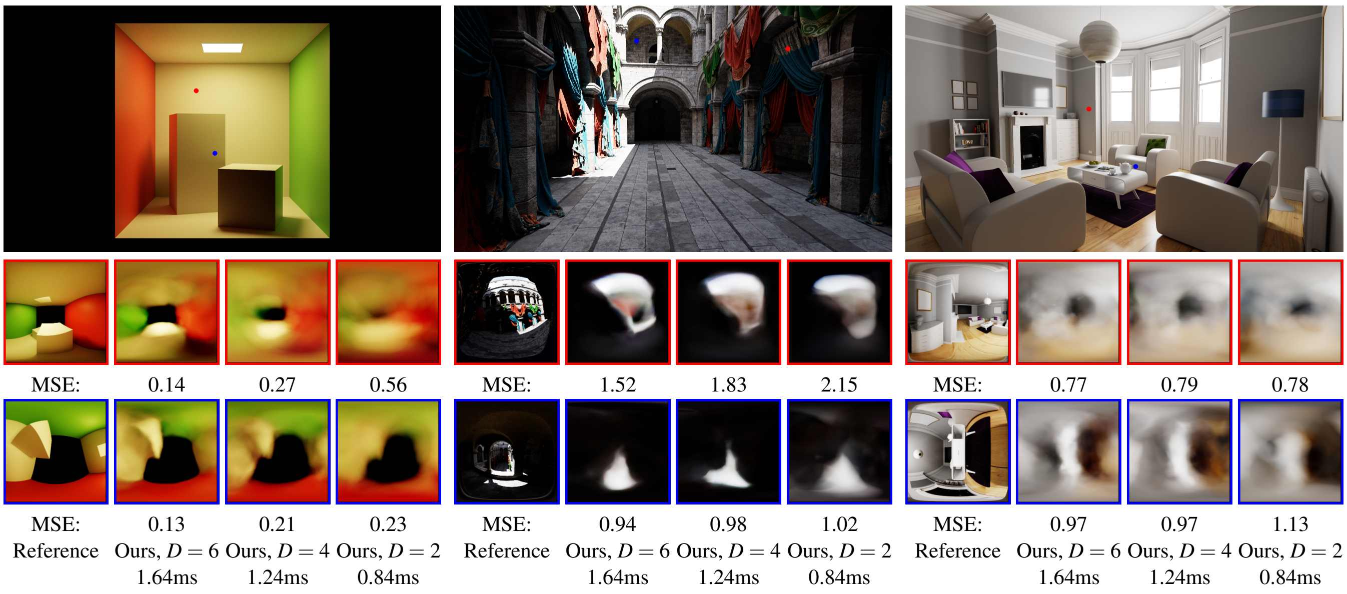


Figure 15: *In this figure, we examine the impact of varying the number of hidden layers D within our* Neural Incident Radiance Cache *(NIRC) on the precision of the reconstructed radiance signal for multiple scenes, arranged from left to right: Cornel Box, New Sponza, and White Room. The HDR cache is showcased for a certain set of points using octahedral projection, and we calculate the* Mean Squared Error *(MSE) between it and the corresponding ground truth reference renders. For better visualization, we convert HDR values to LDR using the ACES tone mapper. Our cache does not store lighting information from emissive surfaces. Additionally, we present the computation time for each additional neural sample per neural architecture, aimed to illustrate scalability and the balance between performance and quality. As observed, a greater number of hidden layers typically result in a substantial improvement in the reconstruction capabilities of the NIRC, although this is not consistently the case across all scenarios.*

tation adds an overhead of 2-3 milliseconds. Implementing encoding evaluations and neural network inference directly in the path tracing kernel might eliminate this overhead and reduce the overall implementation complexity and memory consumption.

8. Conclusion

In this paper, we presented a two-level Monte Carlo estimator for real-time global illumination rendering. We train and evaluate tiny neural networks on the fly to approximate incident radiance, as the first level of the estimator. A second level is used to create an unbiased estimator in the context of multi-level Monte Carlo. We also present a biased variant which can reduce the residual error further in the same rendering time. Both variants produce lower errors in equal time comparisons in most experiments than previously published work on neural radiance caching [MRNK21]. In addition, our experiments revealed that combining the MLMC framework with NIRC can offset integration costs while achieving superior variance reduction of the residual error estimator compared to Control Variates. Our key insights to achieving this are a data representation that stores incident radiance (avoiding a costly ray trace to determine visibility when querying the cache), a carefully selected set of inputs to the neural network (producing more accurate output to support this new storage), and a tightly coupled implementation that avoids round trips to global memory where possible. Our comprehensive cache analysis further demonstrated that we can significantly reduce the number of rays traced for indirect illumination by leveraging our cache, leading to more efficient rendering without significantly sacrificing quality.

We found that the allocation of sample counts for path vertices along a transport path influences the final variance significantly and should be investigated in more detail. This concept aligns with principles previously explored in classical light transport simulations, as demonstrated in the previous works [RGH*22] [VK16].

We only experimented with dynamic scene content but did not specifically optimize the cache to quickly react to drastic changes. For instance, distributing the training workload more into regions of the cache that have low quality might improve the results. Utilizing hierarchical caches has the potential to more quickly distribute information about light sources in the scenes, and could provide interesting interplay with the multi-level Monte Carlo paradigm. We believe that research in this area promises to be applicable for real-time and offline rendering.

9. Acknowledgements

Open Access funding enabled and organized by Projekt DEAL. We also thank all the anonymous reviewers for their insightful comments.

References

[ANA*20] Andersson, Pontus, Nilsson, Jim, Akenine-Möller, Tomas, et al. “FLIP: A Difference Evaluator for Alternating Images”.

Proceedings of the ACM on Computer Graphics and Interactive Techniques 3.2 (2020), 15:1–15:23. DOI: 10.1145/3406183 10.

[BFK18] BINDER, NIKOLAUS, FRICKE, SASCHA, and KELLER, ALEXANDER. "Fast Path Space Filtering by Jittered Spatial Hashing". *ACM SIGGRAPH 2018 Talks*. SIGGRAPH '18. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018. ISBN: 9781450358200. DOI: 10.1145/3214745.3214806 3.

[BMdD*23] BOISSÉ, GUILLAUME, MEUNIER, SYLVAIN, de DINECHIN, HELOISE, et al. *GI-1.0: A Fast and Scalable Two-level Radiance Caching Scheme for Real-time Global Illumination*. 2023. arXiv: 2310.19855 [cs.GR] 2.

[BWP*20] BITTERLI, BENEDIKT, WYMAN, CHRIS, PHARR, MATT, et al. "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting". *ACM Trans. Graph.* 39.4 (July 2020). DOI: 10/gg8xc7 15.

[CDE*14] CIGOLLE, ZINA H., DONOW, SAM, EVANGELAKOS, DANIEL, et al. "A Survey of Efficient Representations for Independent Unit Vectors". *Journal of Computer Graphics Techniques (JCGT)* 3.2 (Apr. 2014), 1–30. ISSN: 2331-7418. URL: http://jcgt.org/published/0003/02/01/ 11.

[CSN*23] CHANG, WESLEY, SIVARAM, VENKATARAM, NOWROUZEZAHRAI, DEREK, et al. "Parameter-space ReSTIR for Differentiable and Inverse Rendering". *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. Los Angeles, CA, USA: Association for Computing Machinery, 2023. DOI: 10.1145/3588432.3591512 15.

[DBN17] DUBOUCHET, RENAUD ADRIEN, BELCOUR, LAURENT, and NOWROUZEZAHRAI, DEREK. "Frequency Based Radiance Cache for Rendering Animations". *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. Ed. by ZWICKER, MATTHIAS and SANDER, PEDRO. The Eurographics Association, 2017. ISBN: 978-3-03868-045-1. DOI: 10.2312/sre.20171193 3.

[DDLF24] DURASOV, NIKITA, DORNDORF, NIK, LE, HIEU, and FUA, PASCAL. "ZigZag: Universal Sampling-free Uncertainty Estimation Through Two-Step Inference". *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856 15.

[DG21] DEUTSCHMANN, NICOLAS and GÖTZ, NIKLAS. *ZüNIS: Normalizing Flows for Neural Importance Sampling*. https://github.com/ndeutschmann/zunis. GitHub repository. 2021 7.

[DPD22] DIOLATZIS, STAVROS, PHILIP, JULIEN, and DRETTAKIS, GEORGE. "Active Exploration for Neural Global Illumination of Variable Scenes". *ACM Trans. Graph.* 41.5 (May 2022). ISSN: 0730-0301. DOI: 10.1145/3522735 3.

[FKH23] FUJIEDA, SHIN, KAO, CHIH-CHEN, and HARADA, TAKAHIRO. *Neural Intersection Function*. 2023. arXiv: 2306.07191 [cs.GR] 3.

[Gil08] GILES, MICHAEL B. "Multilevel Monte Carlo Path Simulation". *Operations Research* 56 (June 2008), 607–617. DOI: 10.1287/opre.1070.0496 3.

[Gil15] GILES, MICHAEL B. "Multilevel Monte Carlo methods". *Acta Numerica* 24 (2015), 259–328. DOI: 10.1017/S096249291500001X 3.

[GKB09] GASSENBAUER, VACLAV, KRIVANEK, JAROSLAV, and BOUATOUCH, KADI. "Spatial Directional Radiance Caching". *Computer Graphics Forum* (2009). ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01496.x 2.

[GKDS12] GEORGIEV, ILIYAN, KŘIVÁNEK, JAROSLAV, DAVIDOVIČ, TOMÁŠ, and SLUSALLEK, PHILIPP. "Light transport simulation with vertex connection and merging". *ACM Trans. Graph.* 31.6 (Nov. 2012). ISSN: 0730-0301. DOI: 10.1145/2366145.2366211 2.

[GSHG98] GREGER, G., SHIRLEY, P., HUBBARD, P.M., and GREENBERG, D.P. "The irradiance volume". *IEEE Computer Graphics and Applications* 18.2 (1998), 32–43. DOI: 10.1109/38.656788 3.

[Hay94] HAYKIN, SIMON. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994 2.

[HCZ21] HADADAN, SAEED, CHEN, SHUHONG, and ZWICKER, MATTHIAS. "Neural radiosity". *ACM Trans. Graph.* 40.6 (Dec. 2021), 1–11. DOI: 10.1145/3478513.3480569 4.

[HOJ08] HACHISUKA, TOSHIYA, OGAKI, SHINJI, and JENSEN, HENRIK WANN. "Progressive photon mapping". *ACM Trans. Graph.* 27.5 (Dec. 2008). ISSN: 0730-0301. DOI: 10.1145/1409060.1409083 2.

[Jen96] JENSEN, HENRIK WANN. "Global Illumination using Photon Maps". *Rendering Techniques '96*. Ed. by PUEYO, XAVIER and SCHRÖDER, PETER. Vienna: Springer Vienna, 1996, 21–30. ISBN: 978-3-7091-7484-5 2.

[Kaj86] KAJIYA, JAMES T. "The Rendering Equation". *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, 143–150. ISBN: 0897911962. DOI: 10.1145/15922.15902 2, 3.

[KB17] KINGMA, DIEDERIK P. and BA, JIMMY. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG] 10.

[KCK*22] KALLWEIT, SIMON, CLARBERG, PETRIK, KOLB, CRAIG, et al. *The Falcor Rendering Framework*. Aug. 2022. URL: https://github.com/NVIDIAGameWorks/Falcor 7, 10.

[KGPB05] KRIVANEK, J., GAUTRON, P., PATTANAIK, S., and BOUATOUCH, K. "Radiance caching for efficient global illumination computation". *IEEE Transactions on Visualization and Computer Graphics* 11.5 (2005), 550–561. DOI: 10.1109/TVCG.2005.83 3.

[LMH*18] LEHTINEN, JAAKKO, MUNKBERG, JACOB, HASSELGREN, JON, et al. "Noise2Noise: Learning Image Restoration without Clean Data". *Proceedings of the 35th International Conference on Machine Learning*. Ed. by DY, JENNIFER and KRAUSE, ANDREAS. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, 2965–2974 5.

[MESK22] MÜLLER, THOMAS, EVANS, ALEX, SCHIED, CHRISTOPH, and KELLER, ALEXANDER. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127 2, 3, 6.

[MGNM19] MAJERCIK, ZANDER, GUERTIN, JEAN-PHILIPPE, NOWROUZEZAHRAI, DEREK, and MCGUIRE, MORGAN. "Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields". *Journal of Computer Graphics Techniques (JCGT)* 8.2 (June 2019), 1–30. ISSN: 2331-7418. URL: http://jcgt.org/published/0008/02/01/ 3.

[Mit87] MITCHELL, DON P. "Generating Antialiased Images at Low Sampling Densities". *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, 65–72. ISBN: 0897912276. DOI: 10.1145/37401.37410 2.

[MJJG18] MARCO, JULIO, JARABO, ADRIAN, JAROSZ, WOJCIECH, and GUTIERREZ, DIEGO. "Second-Order Occlusion-Aware Volumetric Radiance Caching". *ACM Trans. Graph.* 37.2 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3185225 3.

[MMR*19] MÜLLER, THOMAS, MCWILLIAMS, BRIAN, ROUSSELLE, FABRICE, et al. "Neural Importance Sampling". *ACM Trans. Graph.* 38.5 (Oct. 2019), 145:1–145:19. ISSN: 0730-0301. DOI: 10.1145/3341156 2, 4, 7, 15.

[MPC19] MOREAU, PIERRE, PHARR, MATT, and CLARBERG, PETRIK. "Dynamic Many-Light Sampling for Real-Time Ray Tracing". *High-Performance Graphics - Short Papers*. Ed. by STEINBERGER, MARKUS and FOLEY, TIM. The Eurographics Association, 2019. ISBN: 978-3-03868-092-5. DOI: 10.2312/hpg.20191191 10.

[MRKN20] MÜLLER, THOMAS, ROUSSELLE, FABRICE, KELLER, ALEXANDER, and NOVÁK, JAN. "Neural Control Variates". *ACM Trans. Graph.* 39.6 (Nov. 2020), 243:1–243:19. ISSN: 0730-0301. DOI: 10.1145/3414685.3417804 2, 4, 6–8, 15.

[MRNK21] MÜLLER, THOMAS, ROUSSELLE, FABRICE, NOVÁK, JAN, and KELLER, ALEXANDER. "Real-time Neural Radiance Caching for Path Tracing". *ACM Trans. Graph.* 40.4 (Aug. 2021), 36:1–36:16. DOI: 10.1145/3450626.3459812 1–3, 6–8, 10, 13, 14, 16.

[MSS*10] MEYER, QUIRIN, SÜSSMUTH, JOCHEN, SUSSNER, GERD, et al. "On Floating-Point Normal Vectors". *Computer Graphics Forum* 29.4 (2010), 1405–1409. DOI: 10.1111/j.1467-8659.2010.01737.x 11.

[MST*20] MILDENHALL, BEN, SRINIVASAN, PRATUL P., TANCIK, MATTHEW, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". *ECCV*. 2020 4.

[Mül21] MÜLLER, THOMAS. *tiny-cuda-nn*. Version 1.7. Apr. 2021. URL: https://github.com/NVlabs/tiny-cuda-nn 7.

[NRN*23] NICOLET, BAPTISTE, ROUSSELLE, FABRICE, NOVÁK, JAN, et al. "Recursive Control Variates for Inverse Rendering". *ACM Trans. Graph.* 42.4 (Aug. 2023). DOI: 10.1145/3592139 15.

[OZ00] OWEN, ART B. and ZHOU, YI. "Safe and Effective Importance Sampling". *Journal of the American Statistical Association* 95 (2000), 135–143. DOI: 10.1080/01621459.2000.10473909 15.

[Pan20] PANTALEONI, JACOPO. *Online path sampling control with progressive spatio-temporal filtering*. 2020. arXiv: 2005.07547 [cs.GR] 2, 3, 7.

[PGM*19] PASZKE, ADAM, GROSS, SAM, MASSA, FRANCISCO, et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG] 7.

[RBRD22] RAINER, GILLES, BOUSSEAU, ADRIEN, RITSCHEL, TOBIAS, and DRETTAKIS, GEORGE. "Neural Precomputed Radiance Transfer". *Computer Graphics Forum (Proceedings of the Eurographics conference)* 41.2 (Apr. 2022) 3.

[RFGL23] RODRIGUEZ-PARDO, CARLOS, FABRE, JAVIER, GARCES, ELENA, and LOPEZ-MORENO, JORGE. "NEnv: Neural Environment Maps for Global Illumination". *Computer Graphics Forum* 42.4 (2023), e14883. DOI: 10.1111/cgf.14883 3.

[RGH*22] RATH, ALEXANDER, GRITTMANN, PASCAL, HERHOLZ, SEBASTIAN, et al. "EARS: efficiency-aware russian roulette and splitting". *ACM Trans. Graph.* 41.4 (July 2022). ISSN: 0730-0301. DOI: 10.1145/3528223.3530168 16.

[Rud16] RUDER, SEBASTIAN. "An overview of gradient descent optimization algorithms". *arXiv preprint arXiv:1609.04747* (2016) 15.

[RZD14] REHFELD, HAUKE, ZIRR, TOBIAS, and DACHSBACHER, CARSTEN. "Clustered Pre-convolved Radiance Caching". *Eurographics Symposium on Parallel Graphics and Visualization*. Ed. by AMOR, MARGARITA and HADWIGER, MARKUS. The Eurographics Association, 2014. ISBN: 978-3-905674-59-0. DOI: 10.2312/pgv.20141081 3.

[SL17] SILVENNOINEN, ARI and LEHTINEN, JAAKKO. "Real-Time Global Illumination by Precomputed Local Reconstruction from Sparse Radiance Probes". *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130852 3.

[SNRS12] SCHERZER, DANIEL, NGUYEN, CHUONG H., RITSCHEL, TOBIAS, and SEIDEL, HANS-PETER. "Pre-convolved Radiance Caching". *Computer Graphics Forum* 31.4 (2012), 1391–1397. DOI: 10.1111/j.1467-8659.2012.03134.x 3.

[VG95] VEACH, ERIC and GUIBAS, LEONIDAS J. "Optimally Combining Sampling Techniques for Monte Carlo Rendering". *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, 1995, 419–428. ISBN: 0897917014. DOI: 10.1145/218380.218498 5, 10.

[VHH*19] VORBA, JIŘÍ, HANIKA, JOHANNES, HERHOLZ, SEBASTIAN, et al. "Path Guiding in Production". *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. Los Angeles, California: ACM, 2019, 18:1–18:77. ISBN: 978-1-4503-6307-5. DOI: 10.1145/3305366.3328091 2.

[VHM*22] VERBIN, DOR, HEDMAN, PETER, MILDENHALL, BEN, et al. "Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields". *CVPR* (2022) 6.

[VK16] VORBA, JIŘÍ and KŘIVÁNEK, JAROSLAV. "Adjoint-driven Russian roulette and splitting in light transport simulation". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925912 16.

[VKJ19] VICINI, DELIO, KOLTUN, VLADLEN, and JAKOB, WENZEL. "A Learned Shape-Adaptive Subsurface Scattering Model". *ACM Trans. Graph.* 38.4 (July 2019), 15. DOI: 10.1145/3306346.3322974 4.

[VKŠ*14] VORBA, JIŘÍ, KARLÍK, ONDŘEJ, ŠIK, MARTIN, et al. "On-line Learning of Parametric Mixture Models for Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33.4 (Aug. 2014) 2, 7.

[VPG14] VARDIS, KOSTAS, PAPAIOANNOU, GEORGIOS, and GKARAVELIS, ANASTASIOS. "Real-time Radiance Caching using Chrominance Compression". *Journal of Computer Graphics Techniques (JCGT)* 3.4 (Dec. 2014), 111–131. ISSN: 2331-7418. URL: http://jcgt.org/published/0003/04/06/ 3.

[WRC88] WARD, GREGORY J., RUBINSTEIN, FRANCIS M., and CLEAR, ROBERT D. "A Ray Tracing Solution for Diffuse Interreflection". *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '88. New York, NY, USA: Association for Computing Machinery, 1988, 85–92. ISBN: 0897912756. DOI: 10.1145/54852.378490 2, 3.

[XTS*22] XIE, YIHENG, TAKIKAWA, TOWAKI, SAITO, SHUNSUKE, et al. "Neural Fields in Visual Computing and Beyond". *Computer Graphics Forum* (2022). ISSN: 1467-8659. DOI: 10.1111/cgf.14505 4.

[ZBN19] ZHAO, YANGYANG, BELCOUR, LAURENT, and NOWROUZEZAHRAI, DEREK. "View-dependent Radiance Caching". *Proceedings of Graphics Interface 2019*. GI 2019. Kingston, Ontario: Canadian Information Processing Society, 2019. ISBN: 978-0-9947868-4-5. DOI: 10.20380/GI2019.22 3.

[ZRW*23] ZELTNER, TIZIAN, ROUSSELLE, FABRICE, WEIDLICH, ANDREA, et al. *Real-Time Neural Appearance Models*. 2023. arXiv: 2305.02678 [cs.GR] 4.

[ZXS*21] ZHU, SHILIN, XU, ZEXIANG, SUN, TIANCHENG, et al. "Photon-Driven Neural Reconstruction for Path Guiding". *ACM Trans. Graph.* 41.1 (Nov. 2021). ISSN: 0730-0301. DOI: 10.1145/3476828 4.