Neural Quadrature Rule and Autoregressive Adaptive Sampling

HAOLIN LU, University of California San Diego, USA
LIWEN WU, University of California San Diego, USA
ZIMO WANG, University of California San Diego, USA
TZU-MAO LI, University of California San Diego, USA
RAVI RAMAMOORTHI, University of California San Diego, USA

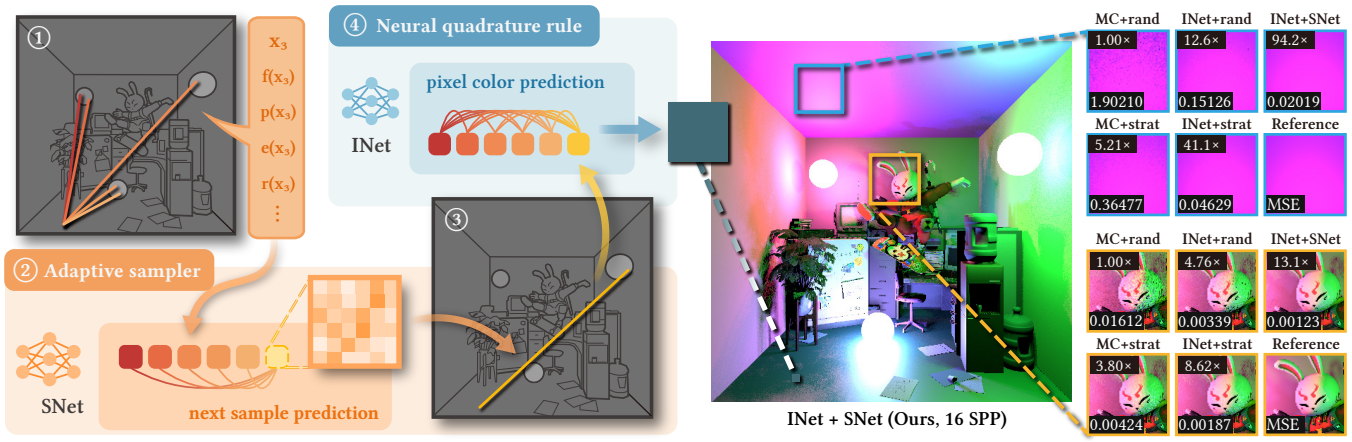


Fig. 1. Our Approach for Direct Illumination. ① For each traced light path, we usually can obtain not only the function value $f(X_i)$, but also rich auxiliary information, such as the sample position X_i and, if the path hits a spherical area light, the light position $p(X_i)$, emission $e(X_i)$, and radius $r(X_i)$, etc. While such information is typically ignored by a standard Monte Carlo estimator, we encode all these features into a sample token. ② Our adaptive sampler decides the direction of the next path, by observing all previous path samples and predicting the most informative direction. This is implemented using causal attention to model the distribution of the next sample. ③ We then draw the next sample from this distribution, trace the corresponding adaptive path, collect its information, and encode it as a new token. ④ After collecting enough samples, we estimate the integral by attending to all sample tokens within the same pixel using bidirectional attention, and output the final pixel color. Our method does not involve any spatial reuse and operates purely as a per-pixel integrator. Our neural sampler and integrator are jointly trained on a large dataset of scenes and runs in a purely feed-forward manner at inference time. With the same number of samples, it achieves significant improvements over Monte Carlo baselines, even on many previously unseen scenes.

Monte Carlo integration is widely used in computer graphics, especially in rendering, but we identify two key limitations. First, for each sample, we can often obtain rich auxiliary information, such as sample position, or geometric information. However, a classical Monte Carlo estimator cannot effectively use this information and only averages the function values. Second, the Monte Carlo formulation makes it difficult to adapt the sampling distribution toward truly informative regions, which can be critical for reconstructing the signal. To address these limitations, we argue that a sampler and integrator beyond the standard Monte Carlo methods is needed. We therefore propose an end-to-end sampling–integration approach that jointly learns both a sampler and an integrator using neural networks, enabling samples to be drawn and used in a more coupled and principled manner. By training on a dataset of integrands, the estimator can further use learned priors over integrand structure and specialize to a family of problems. We evaluate our method on diverse applications in lighting, transmittance, generalized winding number, and walk-on-spheres, spanning both linear and nonlinear cases, and a broad range of low- and high-dimensional settings. Even though the networks add computational overheads, in the equal-sample setting, our method achieves substantial improvements, providing a powerful alternative to traditional quadrature rules and sampling methods.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: integration, reconstruction, transformer, quadrature rule, adaptive sampling, reinforcement learning

Authors' addresses: Haolin Lu, hal128@ucsd.edu, University of California San Diego, USA; Liwen Wu, liw026@ucsd.edu, University of California San Diego, USA; Zimo Wang, zimowang@ucsd.edu, University of California San Diego, USA; Tzu-Mao Li, tzli@ucsd.edu, University of California San Diego, USA; Ravi Ramamoorthi, ravir@cs.ucsd.edu, University of California San Diego, USA.

1 INTRODUCTION

In computer graphics and related fields, integration has been predominantly performed by Monte Carlo[1] and numerical quadrature methods, accompanied by some form of stratified sampling. In this work, we show that there is significant room for improvement we can potentially gain by moving away from the Monte Carlo regime, and jointly considering the sampling and the integration tasks.

Monte Carlo integration methods ignore important information and impose unnecessary constraints if we consider them as a *reconstruction* of the underlying signals for integration [O'Hagan 1987].

[1]While the term Monte Carlo can broadly refer to any stochastic algorithm, in this work we specifically use it to denote Monte Carlo integration methods that rely on expectations and the law of large numbers to guarantee unbiasedness or consistency.

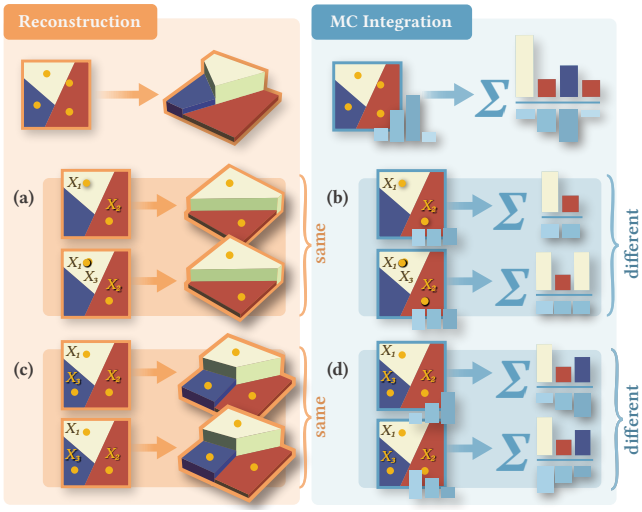


Fig. 2. Mismatch between reconstruction and MC integration. Both reconstruction and integration use point samples to learn about a function. Reconstruction aims to recover the function's shape, while integration aims to estimate its integral. However, Monte Carlo integration sometimes behaves quite differently from reconstruction. (a) Consider three samples X_1, X_2, and X_3, where X_3 is identical to X_1. Since X_3 adds no new information, a reconstruction method would usually behave the same as if only X_1 and X_2 were given. (b) Monte Carlo, however, averages all three samples equally, even though X_1 and X_3 overlap. This happens because Monte Carlo does not use the spatial information between sample positions. (c) As another example, if we obtain three samples X_1, X_2, and X_3, a reconstruction method will usually give the same prediction regardless of how these samples were drawn. (d) Monte Carlo, however, can give different estimates even when the sample sets are exactly identical, if they were drawn using different importance sampling distributions. This happens even though the estimators obtain the same information about the integrand, revealing Monte Carlo's heavy, and sometimes excessive, dependence on the sampling distribution.

For one, they ignore significant auxiliary information such as sample locations (Fig. 2(b)), derivatives, or in the case of light transport, the material, lighting, and geometric information associated with the corresponding light paths. This auxiliary information can provide critical information for reconstructing the latent signal. For another, importance sampling and adaptive importance sampling methods weigh the samples using their probability densities, and encourage the sampling density to be proportional to the integrand. However, importance sampling significantly restricts the algorithm design by making both sampling and integration heavily dependent on the sampling distribution (Fig. 2(d)) and prevents adaptive reconstruction strategies such as sampling close to sharp features or boundaries. While numerical quadrature methods and their stochastic variants [Crespo et al. 2021; Haber 1969; Salaün et al. 2022] partially alleviated some of the issues, they still have to assume particular function classes of the integrand for analytical integration and still significantly constrain the design of the sampling routine.

We broadly generalize existing methods by treating sampling and integration as an end-to-end learning task. Instead of using a hand-designed estimator, we train an integration neural network (INet, Fig. 1 ④) to predict the integral directly from all sample observations, enabling it to exploit auxiliary information and flexible sampling strategies, at the cost of giving up strict unbiasedness and consistency guarantees. We show that a network predicting the weights of samples can subsume most existing methods including numerical quadratures, control variates [Lu et al. 2025; Salaün et al. 2022], and multiple importance sampling [Veach and Guibas 1995]. To complement this learned integrator, we employ a second sampling neural network (SNet, Fig. 1 ②) to decide where to draw the next sample, with the goal of maximizing the integrator's predictive accuracy. We show how to address the lack of direct supervision by learning the sampling distribution in a self-supervised manner using reinfocement learning techniques. Through joint training, the sampler learns where to adaptively draw samples that are most informative for the integrator, while the integrator learns to effectively use the resulting rich observations.

Our results (e.g., Fig. 1) suggest that there is a lot of sampling and integration efficiency left on the table by (quasi) Monte Carlo integration, even *without* reusing sample information across pixels as in denoising. While our approach incurs neural inference overhead and may not yet outperform baselines under equal-time budgets, it shows clear advantages for equal-sample comparisons. We believe this opens a new direction for integration, particularly when sampling is expensive (e.g., complex scenes or CPU rendering) and as neural inference continues to accelerate on modern hardware.

Our technical contributions are:

- To address the fundamental limitations of Monte-Carlo-based estimators, we propose a learning-based integration method that jointly learns sampling and integration, enabling adaptive sampling and effective use of samples from the perspective of sampling and reconstruction.
- We show that viewing integration as sample reweighting subsumes and generalizes existing methods and enables effective network design. Based on this view, we train an integration network (INet) to weigh the samples (Section 4).
- Learning an adaptive sampling distribution is challenging due to the lack of direct supervision data. We design an autoregressive sampling network (SNet) and train it jointly with the integrator (INet) using reinforcement learning algorithms in a self-supervised manner (Section 5).
- We demonstrate that our approach generalizes and provides significant accuracy improvements over Monte Carlo methods in an equal-sample setting. We show a wide range of integration tasks across different dimensionalities, including physically based rendering (1D–6D), generalized winding numbers and smoothed distances of curves (1D), and Walk-on-Spheres (12D) (Section 6 and Appendix E).

An open-source implementation of our method is available under https://github.com/suikasibyl/nqr.

2 RELATED WORK

2.1 Integration and reconstruction

Quadrature rules. Numerical quadrature rules approximate definite integrals by forming weighted averages of sampled function values, with examples including the rectangle rule, the trapezoidal

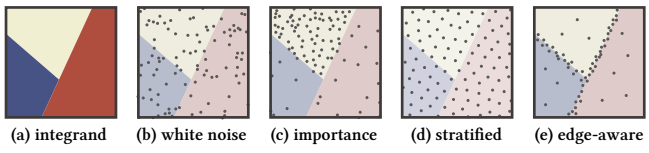


Fig. 3. Established sampling strategies. **(a)** For a given function, different sampling strategies can be used for integration or reconstruction. **(b)** Random white noise can be used in either setting, but it is usually not the most efficient. **(c)** Importance sampling places more samples in high-valued regions, which is beneficial for Monte Carlo integration but less reasonable from a reconstruction perspective. **(d)** Stratified sampling distributes samples more uniformly than white noise and is therefore usually more informative in general. **(e)** Many denoisers prefer to place samples near boundaries, as these regions contain more information about discontinuities. However, this strategy has not yet been shown to benefit Monte Carlo integration, largely due to its overdependence on importance sampling.

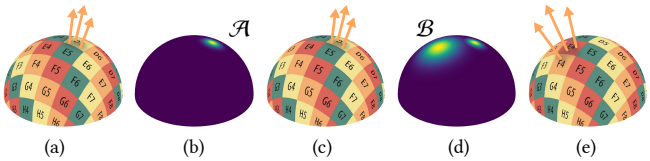


Fig. 4. Paradox in path guiding. **(a)** In path guiding, suppose we have traced samples in region $\mathcal{A}$ and **(b)** identified a high-valued lobe. **(c)** Importance sampling then encourages drawing more samples in $\mathcal{A}$, even though this region is already well explored and provides little new information. **(d)** Meanwhile, another lobe may or may not exist in region $\mathcal{B}$. **(e)** Sampling in $\mathcal{B}$ would greatly reduce uncertainty by revealing whether such a lobe exists. However, although these samples are informative from a reconstruction perspective, they can harm Monte Carlo estimates if $\mathcal{B}$ contains only low values. Ideally, an estimator should benefit from uncertainty reduction regardless of whether the sampled values in $\mathcal{B}$ are high or low.

rule [Quarteroni et al. 2006], and Monte Carlo methods [Owen 2013; Robert et al. 1999]. Many handcrafted quadrature rules can be interpreted as explicitly reconstructing the integrand. For example, the trapezoidal rule corresponds to a piecewise linear reconstruction. Such methods can achieve high accuracy in low-dimensional settings, highlighting the close connection between integration and reconstruction. However, explicit reconstruction becomes exponentially more difficult as dimensionality increases, causing these approaches to quickly suffer from the curse of dimensionality. In such cases, Monte Carlo methods become a practical alternative. Our work suggests that a neural quadrature can improve sample efficiency over handcrafted rules in low dimensions and also offer advantages over Monte Carlo methods in higher-dimensions.

Denoising. Denoising [Li et al. 2012; Moon et al. 2016; Schied et al. 2017; Zwicker et al. 2015] is a prominent example of reconstruction in rendering, where noisy observations are used to recover a clean image. Neural network–based approaches [Bako et al. 2017; Balint et al. 2023; Chen et al. 2024; Kalantari et al. 2015] have shown strong ability to learn effective reconstruction strategies that often outperform manually designed methods. This success motivates us to adopt neural techniques for integration as well. Some extensions of denoising methods incorporate high-dimensional integrand spaces [Hachisuka et al. 2008; Sen et al. 2011] as well as sample- or path-space information [Cho et al. 2021; Gharbi et al. 2019; Lin et al. 2021], which are more closely related to our approach. Our work shows that even without spatial reuse, improved integral estimates can be obtained purely from existing samples. Moreover, our formulation generalizes beyond rendering applications.

Reconstruction for Monte Carlo variance reduction. Reconstruction can also serve as an auxiliary tool for MC variance reduction. Path guiding [Bako et al. 2019; Dahm and Keller 2017; Lafortune and Willems 1995; Lu et al. 2024; Müller et al. 2017; Zeng et al. 2025b] reconstructs incoming radiance to guide importance sampling, while many other methods employ reconstructed integrands as control variates [Crespo et al. 2021; Müller et al. 2020; Salaün et al. 2022; Xu and Wang 2025]. These approaches typically rely on explicit, per-scene reconstruction and are designed to preserve unbiasedness, which in turn requires representations that are analytically integrable or sampleable [Li et al. 2024; Meister and Harada 2025; Müller et al. 2018; Müller et al. 2020; Ott et al. 2023; Wu et al. 2025]. Our method avoids explicit reconstruction and per-scene fitting, and relaxes the unbiasedness requirement for a broader design space. Moreover, our approach naturally supports integration with adaptive sampling, which is not addressed in this line of work.

2.2 Sampling strategies

Sampling plays a key role in both integration and reconstruction, as it directly determines where and what information is collected. As shown in Fig. 3, while uniform random sampling is easy to generate, many more advanced strategies have been explored in the literature.

Importance sampling. As one of the most widely used variance reduction techniques, importance sampling places more samples in regions where the function has higher values. Its benefit comes from dividing the function value by the sampling density, which ideally makes the integrand effectively more uniform. However, outside the Monte Carlo context, placing samples in high-value regions can be questionable, since it does not necessarily make the samples more informative. This issue becomes more pronounced in path guiding, where reconstruction is also involved. As illustrated in Fig. 4, importance sampling tends to place more samples in high-valued regions, often leading to repeated sampling of regions that are already well explored. In contrast, exploring unknown regions can significantly reduce uncertainty about the integrand, but may harm Monte Carlo estimators if the new samples have low values. To sidestep this paradox, we really need an integrator that benefits from informative samples rather than high-valued ones.

Quasi-Monte Carlo point sets and sequences. Quasi-Monte Carlo methods use sample sets or sequences that cover the space more uniformly [Ahmed and Wonka 2021; Ahmed et al. 2016; Durand 2011; Niederreiter 1992; Owen 1997a,b; Perrier et al. 2018], by imposing low-discrepency, blue noise properties, or their combination. More recently, differentiable and neural methods have also been used to design sample sets that satisfy multiple criteria [Doignies et al. 2023, 2024; Leimkühler et al. 2019]. Although these point sets are proven

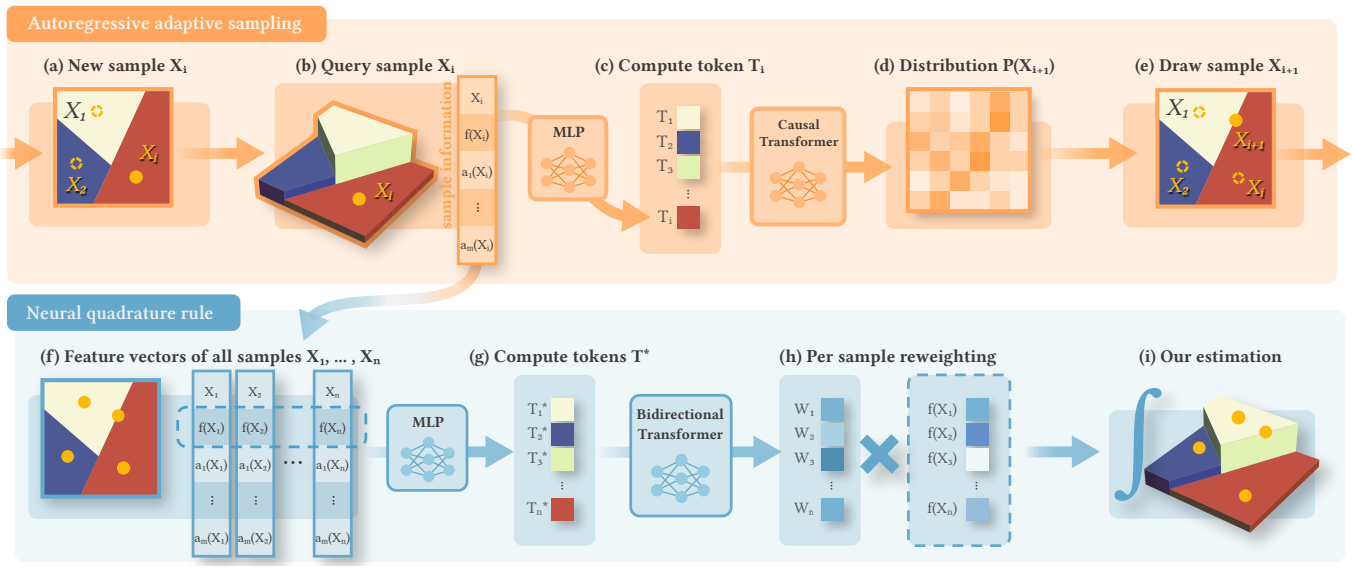


Fig. 5. Our integration pipeline. We estimate an integral by first drawing samples in an autoregressive and adaptive manner to gather information about the integrand (top row). Once all samples are collected, a neural network predicts a reweighting factor for each sample, and the final estimate is computed as a weighted average (bottom row). **(a)** For each new sample X_i, **(b)** we evaluate the function value $f(X_i)$ and collect auxiliary information to form a sample information vector. **(c)** The vector is mapped to a sample token T_i using a multi-layer perceptron (MLP), which is further processed by a causal transformer together with previous tokens, where each token can only attend to earlier ones. **(d)** The transformer predicts a distribution over the next sample location. **(e)** A new sample is drawn from this distribution, and the process repeats until the desired number of samples is obtained. For integration, **(f)** all sample information vectors are re-encoded using a separate MLP. **(g)** The resulting tokens are fed into a bidirectional transformer, where all tokens can attend to each other. **(h)** The transformer predicts a reweighting factor for each sample. **(i)** Finally, the integral is estimated as a reweighted average of all samples.

to be effective, they are generally 1) integrand-agnostic and 2) forced to be uniform, largely because they are designed for the Monte Carlo framework. Instead, we draw samples progressively [Christensen et al. 2018] and aim to use previous samples as observations of integrands to obtain posterior and integrand-aware sampling.

Adaptive sampling. Adaptive sampling [Zwicker et al. 2015] is widely used in denoising to identify samples that are most helpful for reconstruction quality. A-priori methods [Durand et al. 2005] use frequency analysis of rendering behavior to predetermine sample positions, while a-posteriori methods [Hachisuka et al. 2008; Overbeck et al. 2009] analyze existing samples to decide where additional samples should be placed. Typically, the adaptive sampling operate in image space [Firmino et al. 2023; Salehi et al. 2022], with some rare exceptions [Hachisuka et al. 2008]. More recently, learning-based approaches [Firmino et al. 2023; Scardigli et al. 2023] have been explored to optimize reconstruction in an end-to-end manner. Adaptive sampling has also been applied beyond denoising [Huo et al. 2020; Schwarzhaupt et al. 2012; Ward et al. 1988; Xu et al. 2018], but these methods are still developed primarily in the context of reconstruction. In contrast, our method performs sampling and integral estimation within each pixel. These two approaches operate at different stages and can be combined: denoisers guide which pixels to sample, while our method improves estimation within selected pixels. Our approach may introduce artifacts, suggesting joint denoiser training.

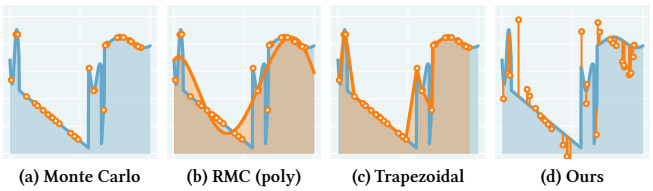
3 OVERVIEW

We propose solving the integration problem using an end-to-end neural approach, as illustrated in Fig. 5. The approach consists of two components: adaptively drawing samples to gather informative observations, and using these observations to predict the integral. In the sampling stage, we collect samples from the integrand in an autoregressive and adaptive manner. To determine where to draw the next sample, we condition on all previous sample information and let a neural network select the most informative location. This process is repeated until the desired number of samples is collected. In the integration stage, all samples and their corresponding auxiliary information are jointly processed by a transformer [Vaswani et al. 2017] to predict a reweighting factor for each sample. The integral is then estimated as a weighted average of the sample values.

Our design addresses the aforementioned limitations by learning both how to use auxiliary information and how to perform integration. The learned integrator can leverage priors over integrands and, more importantly, compensate for learned biased sampling (e.g., edge-aware patterns). This removes strict importance sampling constraints and enables adaptive sampling for integration. All benefits come at the cost of bias and additional network evaluation overhead.

4 NEURAL QUADRATURE RULE

In this section, we consider a simplified setting in which the sampling strategy is fixed, and the goal is to predict the integral as accurately as possible given all available sample information. As a



(a) Monte Carlo (b) RMC (poly) (c) Trapezoidal (d) Ours

Fig. 6. 1D QUADRATURE RULES. Several rules can be applied: **(a)** Basic Monte Carlo estimates the integral by uniformly averaging all sample values. **(b)** Regression-based Monte Carlo (RMC) [Salaün et al. 2022] fits a polynomial approximation to the samples and uses it as a control variate. **(c)** The trapezoidal rule connects adjacent samples to form a piecewise linear approximation and computes its analytic integral. **(d)** Our method reweights each sample value and estimates the integral as a weighted average.

Table 1. MSE AND VARIANCE OF 1D RULES. We report MSE and variance for 1D quadrature rules with $N = 32$ samples, averaged over five test integrands (see Table 7), along with improvement ratios over standard MC. POLY-3 and POLY-5 denote regression Monte Carlo with degree-3 and degree-5 polynomials, and TRAPZ the trapezoidal rule. Our integrator-only approach achieves orders-of-magnitude MSE improvements over standard MC.

Case	MC	Trapz	Poly-3	Poly-5	Ours	ratio
avg	1.47e-2	3.98e-3	1.35e-2	3.04e-2	**6.18e-4**	23.7×
	1.47e-2	3.95e-3	1.32e-2	3.03e-2	**5.13e-4**	28.6×

motivating example, we study the integration of a family of one-dimensional piecewise-continuous functions, as shown in Fig. 3[2], and assume all samples are drawn using uniform random sampling.

To solve this problem, we can use several existing approaches (Fig. 6): (1) basic Monte Carlo, which simply averages all sample values; (2) Monte Carlo methods augmented with reconstruction, such as regression-based Monte Carlo (RMC) [Salaün et al. 2022]; and (3) classical quadrature such as the trapezoidal rule[3].

While widely used, these methods have important limitations: (1) they are designed for arbitrary functions and do not specialize to the piecewise-continuous integrands considered here; (2) they do not leverage available auxiliary information, such as the segment indicator I_i, which existing estimators cannot easily exploit.

Our solution. To address this issue, we use a neural network (Fig. 7) to predict a reweighting factor for each sample. Trained on a large dataset of such integrands, it learns to specialize the estimator for this function class and leverage auxiliary information.

We use transformer blocks [Vaswani et al. 2017] to handle a varying number of samples, making the estimator as flexible as Monte Carlo methods. In contrast, many classical quadrature rules require an exponential growth in the number of samples as dimensionality increases. However, we require storing all samples ($O(N)$ memory).

[2]For compactness, detailed descriptions of all experimental setups are deferred to the appendix.

[3]Although the trapezoidal rule is typically defined on uniform grids, we generalize it to take random samples for a fair comparison, as defined in Appendix B.1.

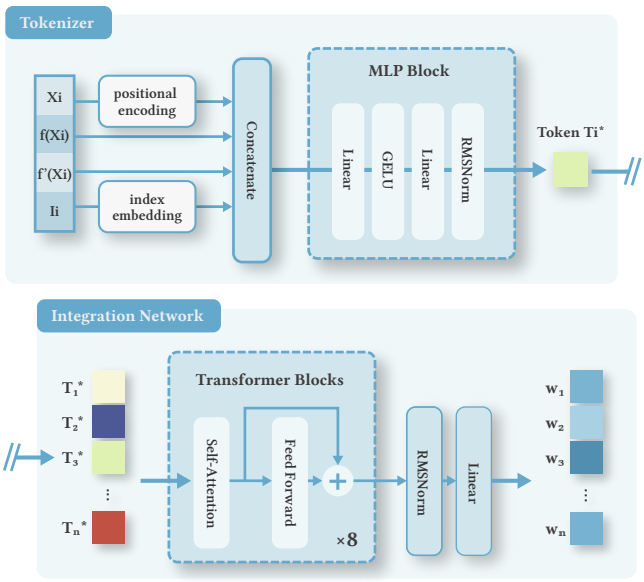


Fig. 7. INTEGRATION NETWORK ARCHITECTURE. Our integrator first tokenizes each sample information vector. Positional information, such as the sample location X_i, is encoded using Fourier positional encoding [Mildenhall et al. 2020], while discrete index information, such as the piece index I_i, is embedded using a learnable embedding lookup. These features are concatenated and passed through an MLP to produce a sample token T_i^*. All tokens are then processed by a stack of bidirectional transformer blocks and finally projected to a reweighting value w_i for each sample.

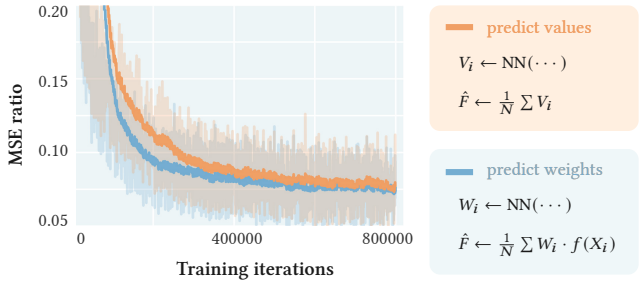


Fig. 8. PREDICTING WEIGHTS OR VALUES. We compare two designs: directly predicting per-sample values and estimating the integral by averaging these values, and predicting per-sample weights and estimating the integral as a weighted average. Both designs are trained for 800,000 iterations. Predicting weights consistently outperforms predicting values, confirming that weighted averaging provides a reasonable inductive bias for integration. The MSE ratio reports the mean squared error of our prediction normalized by the MSE of simple Monte Carlo.

Why weighted averages? We predict a reweighting factor W_i for each sample using a neural network, and estimate the integral F as:

$$\hat{F} = \frac{1}{N}\sum_{i=1}^{N} W_i \cdot f(X_i) \tag{1}$$

But why not let the network predict the integral value directly?

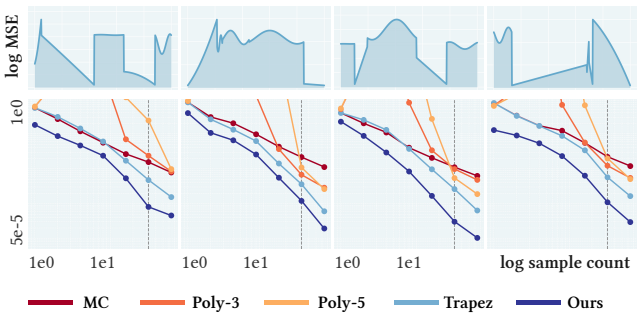


Fig. 9. Convergence of 1D rules. We plot convergence curves for all quadrature methods on four test integrands, with mean squared error (MSE) shown as a function of sample count in log–log scale. Our method consistently outperforms all baselines across different sample counts and typically exhibits super-linear convergence. The dashed line marks $N = 32$ samples, which corresponds to the sample count used for supervision during training.

Our design is motivated by the observation that many existing quadrature rules and Monte Carlo variance reduction methods can be interpreted as sample reweighting. Examples include the rectangle and trapezoidal rules, Gauss–Legendre quadrature, importance sampling, stratification, difference and ratio control variates [Lu et al. 2025], and multiple importance sampling (MIS) [Veach 1998], with detailed derivations provided in Appendix B. We therefore hypothesize that weighted averaging provides an effective inductive bias for the integration task, and the hypothesis is validated by empirical ablation results, as shown in Fig. 8.

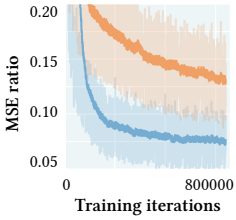
Training. We train the integrator by minimizing the mean squared error between the predicted integral and a reference value:

$$\mathcal{L} = \mathrm{MSE}(\hat{F}, F_{\mathrm{ref}}). \tag{2}$$

The reference integral F_{ref} is computed using Monte Carlo estimation [Lehtinen et al. 2018] with a large number of stratified samples; in the 1D experiments, we use 512 samples. Although the integrator can be trained with varying sample counts N, for simplicity we use a fixed sample count of $N = 32$. At each training iteration, we randomly generate 256 integrand functions. We optimize the network using AdamW for 800,000 iterations, as shown in Fig. 8.

Toy results. Our trained neural quadrature rules significantly outperform existing methods under equal-sample comparisons, as reported in Table 1. The predictions exhibit substantially lower variance than other approaches, at the cost of a small bias, resulting in orders of magnitude improvement in mean squared error (MSE). Note that both bias and variance are jointly minimized during training through the MSE objective. Moreover, although the model is trained using a fixed sample count of $N = 32$, it may also generalize to other sample counts, as shown in Fig. 9. Our method consistently outperforms all baselines across a wide range of sample counts, including $N = 1$, where the learned prior over functions plays a crucial role, and $N = 64$, which was not seen during training.

To verify that auxiliary information contributes to estimation, we perform an ablation study as shown in in Fig. 10, where we train a



	MC	Trapz	w/o Aux	w/ Aux
6	2.2e-2	2.9e-3	2.2e-3	**7.9e-4**
7	1.2e-2	2.6e-3	8.7e-4	**4.1e-4**
8	2.4e-2	6.5e-3	2.7e-3	**1.1e-3**
9	1.5e-2	5.0e-3	1.5e-3	**1.1e-3**
10	8.7e-3	2.8e-3	8.8e-4	**3.8e-4**

Fig. 10. Ablation without auxiliary information. (**Left**) Training loss curves comparing the full auxiliary input setting—including sample positions X_i, function values $f(X_i)$, gradients $f'(X_i)$, and indicators I_i (**blue**)—against a reduced setting that uses only X_i and $f(X_i)$ (**orange**). (**Right**) Mean squared error (MSE) on five test functions, where the best result is shown in **bold** and the second best is underlined. Even without additional auxiliary, our integrator outperforms the trapezoidal rule.

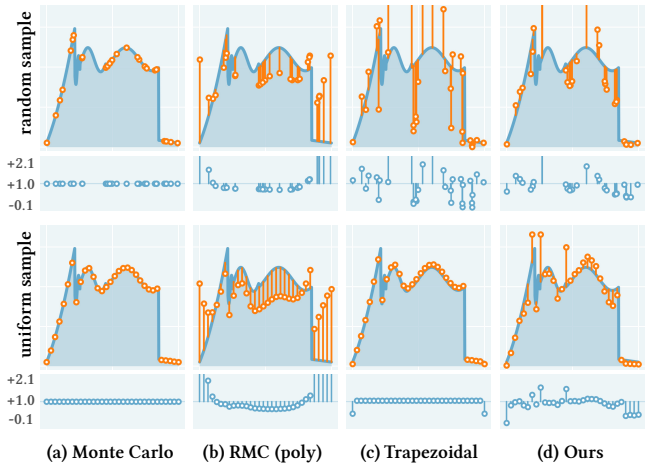


Fig. 11. Reweighting scheme visualization. We visualize how different quadrature rules reweight samples using random samples (top row) and uniform-grid samples (bottom row). Orange circles indicate the reweighted sample values, while blue circles show the absolute reweighting factors.

variant of our model that does not take the gradient $f'(X_i)$ and the segment indicator I_i as input. Removing this auxiliary information leads to a noticeable drop in accuracy compared to the full model. Nevertheless, this variant still outperforms the trapezoidal rule, even though both methods rely on the same sample information: the sample locations X_i and values $f(X_i)$. This result indicates that our approach can still leverage learned priors over the specific family of integrands, even in the absence of additional auxiliary information. We do another experiment (Table 8) by randomizing auxiliary inputs, confirming that the model actively exploits this information.

Reweighting schema. We also reformulate regression-based Monte Carlo and the trapezoidal rule as weighted averages, as derived in the Appendix B, and provide a side-by-side comparison with our learned reweighting in Fig. 11. Basic Monte Carlo simply averages all sample values and applies no reweighting. In contrast, regression-based Monte Carlo explicitly reweights samples to make their contributions more uniform: samples with larger values receive weights

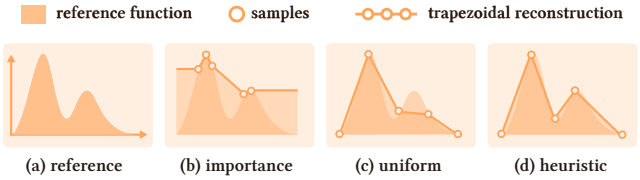


Fig. 12. Effect of sampling on the trapezoidal rule. We study how different sampling strategies affect **(a)** the estimation of a one-dimensional integral using the trapezoidal rule with five samples. **(b)** Importance sampling concentrates samples in high-valued regions, which can lead to a poor and overestimated reconstruction. **(c)** Uniform-grid sampling is commonly used in quadrature rules, but may still miss important details at low sample counts. **(d)** Ideally, samples should be distributed to better capture the function shape, leading to a more accurate piecewise-linear approximation.

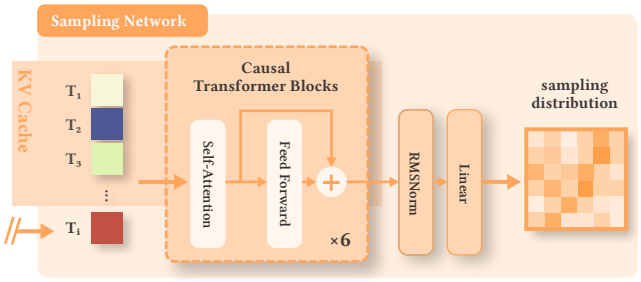


Fig. 13. Sampling network architecture. Our sampler first tokenizes each sample information vector using the same tokenizer architecture as in Fig. 7. The most recent sample token then attends to all previously generated tokens stored in a KV cache through causal attention in a decoder-only transformer, which outputs a sampling distribution for the next sample.

smaller than one, while those with smaller values receive larger weights. This behavior is shared by many variance reduction techniques, including importance sampling and control variates, whose goal is to reduce variance by making each term in the average as uniform as possible. The trapezoidal rule, on the other hand, reweights samples based on the distances to their neighbors: samples with closer neighbors are downweighted, while those with more distant neighbors receive larger weights. In the extreme case of uniformly spaced samples, shown in the second row, it applies equal weights and thus behaves almost similarly to basic MC.

Our learned reweighting shows a trend similar to the trapezoidal rule, but with more subtle adjustments. Unlike many variance reduction methods, it does not force all samples toward a common mean. This is because our weights depend on the full set of samples $X_{1:N}$, which allows a much richer class of weighting strategies. In contrast, many classical methods essentially aim for $W_i^\star = F/f(x_i)$. We discuss this difference in more detail in Appendix C.1.

5 AUTOREGRESSIVE ADAPTIVE SAMPLING

While our neural integrator already achieves substantial improvements, further gains can be obtained by improving the sampling strategy. We motivate our discussion by illustrating how different sampling strategies affect the trapezoidal rule, as shown in Fig. 12. Here, importance sampling can make the trapezoidal estimate prone to overestimation, while stratification is also not optimal since it does not adapt to the integrand. Ideally, sampling should be both integrand-aware and integrator-aware, distributing samples to maximize the information gained for the integral, similar to how adaptive sampling is used in reconstruction.

Why autoregressive sampling. Zwicker et al. [2015] categorize existing adaptive sampling strategies into a-priori and a-posteriori methods, and we aim to combine the strengths of both. By training the sampler on a dataset of integrand functions, it can naturally learn a prior over this function family. To also use posterior information about the specific integrand being evaluated, the sampling process should condition on previous sample observations.

Many existing methods adopt a multi-stage design: they first draw N_{init} samples to gather initial information, then draw N_{adapt} adaptive samples based on these observations, and may repeat this process multiple times [Hachisuka et al. 2008; Huo et al. 2020; Overbeck et al. 2009]. However, samples within the same stage do not benefit from information provided by other samples in that stage, and the hyperparameters N_{init} and N_{adapt} must be manually tuned.

We instead propose drawing adaptive samples in an autoregressive manner, where each new sample conditions on all previously observed samples. A representative example of this design is Bayesian quadrature [Ghahramani and Rasmussen 2002; O'Hagan 1991], which reconstructs the integrand in a probabilistic manner. Osborne et al. [2012] use a Gaussian process prior to guide uncertainty-driven sampling, conditioning on all previous samples (see Fig. 43 and Appendix B.1). However, its smoothness assumption fails on many practical problems, including our piecewise-smooth integrands (Table 6). However, without Gaussian priors, uncertainty becomes intractable, motivating our end-to-end solution.

Our solution. We draw inspiration from the next-token prediction paradigm [Radford et al. 2019] in large language models, which generates sequences autoregressively by conditioning on all previous tokens. Likewise, we model adaptive sampling by predicting each new sample conditioned on all past observations. Accordingly, we use a decoder-only Transformer with causal attention to predict the next sample (Fig. 13), enabling the use of standard techniques such as key–value caching [Vaswani et al. 2017] for faster inference.

Training. While our architecture is highly similar to that of language models, training it is substantially more challenging for several reasons:

(1) *Lack of reference data.* There is no ground-truth optimal sampling sequence for a given integrand, making supervised training of the sampler infeasible.
(2) *Coupling between sampling and integration.* The optimal sampling strategy depends strongly on the integrator. For example, importance sampling is effective for Monte Carlo integration but performs poorly for trapezoidal rules. Conversely, the integrator must also be sampling-aware, as different sampling patterns (e.g., uniform versus edge-aware) require different reweighting behaviors. As a result, the sampler and integrator should be optimized jointly.

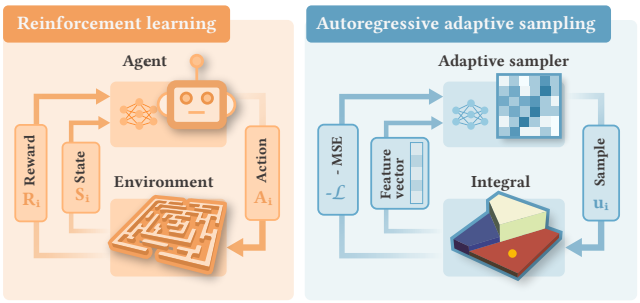


Fig. 14. Autoregressive adaptive sampling as reinforcement learning. This side-by-side comparison highlights the close relationship between our autoregressive adaptive sampling and classical reinforcement learning. In this analogy, the adaptive sampler acts as an agent that learns a policy, where each action corresponds to selecting the position of the next sample. The action interacts with the environment by querying the integrand, producing a new sample information vector that serves as the updated state, while the negative mean squared error of the integral estimate provides the reward. As in standard RL, this sampling process does not assume differentiability of the environment, which enables broad applicability across different tasks.

(3) *Stochasticity and non-differentiability.* The sampler predicts a sampling distribution and draws samples stochastically, which introduces randomness. These samples are then evaluated by querying the integrand, which is often not differentiable, especially in the presence of discontinuities. Both stochastic sampling and non-differentiability prevent direct optimization using standard automatic differentiation.

Challenges (1) and (2) suggest that the entire integration pipeline, as shown in Fig. 5, should be trained end-to-end by minimizing the mean squared error between the predicted integral and a reference value. However, challenge (3) prevents us from directly optimizing the sampler using automatic differentiation. Inspired by recent advances in fine-tuning large language models with reinforcement learning (RL) [Ouyang et al. 2022], we propose to train our sampler using RL. As illustrated in Fig. 14, the adaptive sampling problem naturally aligns with the reinforcement learning formulation.

Specifically, we adopt *Group Relative Policy Optimization* [Shao et al. 2024] (GRPO). Intuitively, at each training iteration, we sample a group of sampling sequences from the current sampler and evaluate their corresponding mean squared errors. Some sequences perform better than others, and GRPO constructs a relative advantage signal within the group to encourage the sampler to generate better-performing sequences while discouraging worse ones. This approach actively explores effective sampling strategies without requiring reference sample sequences, thereby addressing challenge (1). Moreover, it avoids differentiating through stochastic sampling and integrand queries, which sidesteps challenge (3). GRPO is not the only option; we adopt it for generality and simplicity. PPO [Schulman et al. 2017] is also viable but requires an additional critic. Differentiable rendering with reparameterization [Bangaru et al. 2020] is another alternative; however, our problem involves an $N \times D$ integral (Eq. 46) with conditional high-dimensional discontinuities

```python
def joint_training():
  # 1) Roll out G trajectories per integral.
  #    >> x: sample locations
  #    >> y: f(x) values
  #    >> a: auxiliary features
  #    >> ref: reference integral
  #    All with size [Batch, Group, N, ...]
  x, y, a, ref = rollout()

  # 2) Train integrator (INet)
  w   = INet(x, y, a)
  est = mean(w * y, dim=N)      # per (B,G)
  mse = (est - ref)**2          # per (B,G)
  inet_loss = mean(mse)

  inet_optimizer.zero_grad()
  inet_loss.backward()
  inet_optimizer.step()

  # 3) Group-relative advantage
  r   = -detach(mse)
  mu  = mean(r, dim=G)
  sig = std(r, dim=G)
  adv = (r - mu) / (sig + eps)

  # 4) Train sampler (SNet)
  #    REINFORCE update
  logp = SNet.log_prob(x, y, a)
  snet_loss = -mean(logp * adv)

  snet_optimizer.zero_grad()
  snet_loss.backward()
  snet_optimizer.step()
```

Listing 1. Our joint training pseudocode. We first roll out groups of sample sequences using the current sampler. For each sequence, we compute the mean squared error (MSE) of the integrator (INet) estimate and optimize the integrator using supervised learning. The negative MSE is then used as a reward signal, and applied to optimize the sampler (SNet) using GRPO.

Table 2. MSE of 1D Rules and Samplers. We report the MSE of 1D quadrature rules with $N = 32$ samples, using uniform random sampling, stratified sampling, and our joint adaptive sampling, averaged over five scenes (see Table 9). Our jointly trained sampler and integrator reduce the MSE by up to three orders of magnitude compared to standard Monte Carlo methods.

					Ours	
	MC	Trapz	Poly-3	Poly-5	INet	SNet+INet
avg	1.62e-2	4.00e-3	1.25e-2	3.48e-2	7.52e-4	**8.4e-6**
	1.20e-3	9.52e-4	1.17e-3	1.16e-3	2.74e-4	1940×

and additional non-rendering tasks that are not readily supported off the shelf.

While the sampler is optimized using GRPO, the integrator can still be trained with standard supervised learning. Together, this forms a hybrid joint optimization pipeline, as described in Listing 1. To start with a reasonable initial integrator, we first train it separately using uniformly random samples, as described in Section 4.

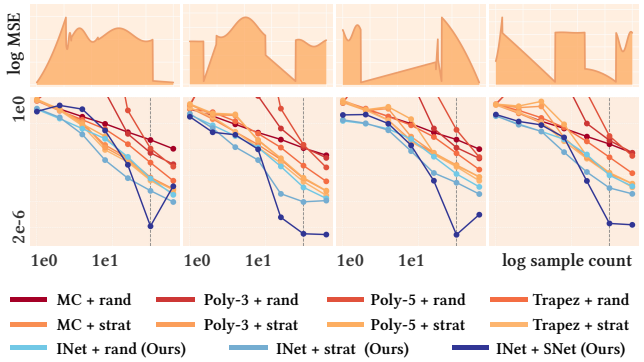


Fig. 15. Convergence of 1D rules with different samplers. We plot convergence curves for all quadrature methods under different sampling strategies on four test integrands, with mean squared error (MSE) shown as a function of sample count in log–log scale. Our jointly trained sampler and integrator consistently outperform all baselines across a wide range of sample counts. The dashed line marks $N = 32$ samples, corresponding to the sample count used for supervision during training.

Toy results. By jointly training the sampler and the integrator, our estimator can further improve integration accuracy. In Table 2, we compare our joint estimator with existing quadrature rules using both random and stratified sampling. Our approach significantly outperforms all baselines and achieves up to three orders of magnitude improvement over basic Monte Carlo with random sampling.

In Fig. 15, we show convergence curves across different sample counts. As the training objective focuses on minimizing the MSE at $N = 32$, the joint estimator performs particularly well at this sample count; performance at other values of N could likely be further improved by supervising multiple sample counts during training. Importantly, our learned integrator alone consistently outperforms other combinations and naturally benefits from stratified sampling, despite being trained only with uniform random samples.

Sampling and weighting schema. In Fig. 16, we visualize the autoregressive sampling behavior of our sampler and the corresponding reweighting produced by the integrator. Overall, the learned strategy can be interpreted as follows: the sampler first places samples near domain boundaries, then explores the domain in a roughly stratified manner. After approximately 12 samples, it begins to actively sample near detected discontinuities in order to more precisely localize their positions, resulting in denser sampling around discontinuities.

The integrator naturally assigns lower weights to samples near discontinuities, which is intuitive given that these regions are more densely sampled by the sampler, while samples in smooth regions receive higher weights as they are more sparsely distributed. This coupled design is enabled only by joint training: the integrator learns how the sampler distributes samples, and the sampler in turn learns how the integrator leverages the collected information.

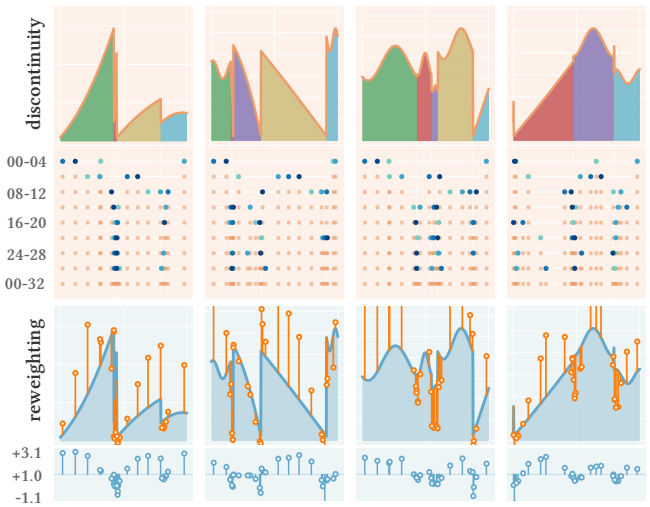


Fig. 16. Learned sampling and reweighting scheme. **(1st row)** Discontinuity structure of four test integrands. **(2nd row)** Each sub-row visualizes the progressive sampling process of our autoregressive sampler. Rows are organized by sampling order: the first sub-row shows samples 0–4, the second shows samples 4–8, and so on, until the final sub-row shows all 32 samples. Newly generated samples in each sub-row are highlighted, while previously drawn samples are shown in light orange to provide context. We observe that the sampler initially explores boundary regions, then distributes samples more uniformly, and finally concentrates samples near discontinuities. **(3rd row)** Reweighted sample values, and **(4th row)** predicted reweighting factors for each sample. Generally, samples near discontinuities are downweighted, while samples in smooth regions receive higher weights.

6 APPLICATIONS

We now show that our method generalizes to a wide range of integration tasks, including generalized winding numbers, smoothed distance and transmittance, direct illumination in physically based rendering, and walk-on-spheres. These experiments span non-negative and real-valued integrals, linear and nonlinear functions, and both low- and high-dimensional settings, demonstrating the generality of our method as a unified integration framework.

Implementation. We implement all experiments in PyTorch [Paszke et al. 2019], with custom GPU kernels written in SlangTorch [Bangaru et al. 2023; He et al. 2018], and perform path tracing using a custom Vulkan-based renderer. All training and inference run on a single GPU (RTX 4090 or RTX 5090). Attention layers are accelerated with FlashAttention2 [Dao 2024] and use BF16 precision.

Equal-sample comparison. We report results under an equal-sample setting. As our method incurs additional cost from neural network evaluation, it is not directly comparable to Monte Carlo under equal-time budgets, given sample queries are much cheaper. Runtime performance is discussed in Section 7 and Appendix F.

6.1 Light transport, direct illumination

To illustrate how our approach applies to practical rendering, we demonstrate its use on direct illumination as a prototype example.

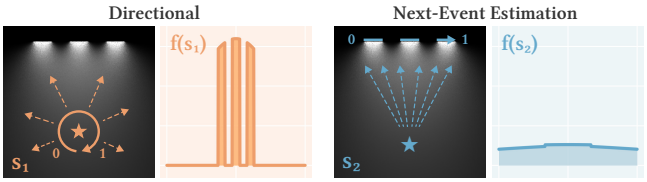


Fig. 17. Importance sampling simplifies integrands. We consider a simple 2D diffuse light transport example, where we compute irradiance in a scene lit by three area lights. With a directional parameterization, the integrand exhibits six discontinuities and strong variations, making it harder to solve. Using next-event estimation, the integral is reparameterized onto the area light segments. In the new sample space s_2, the integrand has only two discontinuities and is much smoother. Intuitively, integrands of the form $f(s_2)$ are easier to integrate than $f(s_1)$, regardless of the integration method.

Dataset. Since our approach requires training on a large dataset of target integrals, we synthesize a dataset of toy rendering scenes, as illustrated in Fig. 39. The training set contains 50,000 scenes.

Importance sampling. Although we previously regarded importance sampling as a less ideal strategy for drawing samples, this does not imply that existing variance-reduction techniques should be discarded. On the contrary, many of them can be naturally combined with our approach and can further enhance its performance. The key observation is that importance sampling can be viewed as a reparameterization of the integral, which often results in a smoother integrand. For example, as illustrated in Fig. 17, next-event estimation (NEE) produces a much simpler integrand in the resulting primal sample space (PSS). Since our method also performs better if the target integrand family is easier to handle, we thus apply it on top of the PSS induced by NEE and BSDF importance sampling.

Multiple importance sampling. Since we still use both NEE and BSDF importance sampling, a natural question is how to combine them into a single estimator. In Monte Carlo rendering, this is typically achieved using multiple importance sampling (MIS):

$$\hat{F}_{\text{MIS}} = \frac{1}{N}\sum_{i=1}^{N} w_1\left(X_i^1\right)\frac{f\left(X_i^1\right)}{p_1\left(X_i^1\right)} + \frac{1}{N}\sum_{i=1}^{N} w_2\left(X_i^2\right)\frac{f\left(X_i^2\right)}{p_2\left(X_i^2\right)}, \quad (3)$$

where X_i^1 and X_i^2 are samples drawn from the two techniques, and w_1, w_2 are the corresponding MIS weights. As discussed in Appendix B.2, MIS is itself a form of reweighting, which allows it to be seamlessly integrated into our method, with the formula:

$$\hat{F}_{\text{ours}} = \frac{1}{N}\sum_{i=1}^{N} W_1\frac{f\left(X_i^1\right)}{p_1\left(X_i^1\right)} + \frac{1}{N}\sum_{i=1}^{N} W_2\frac{f\left(X_i^2\right)}{p_2\left(X_i^2\right)}, \quad (4)$$

where W_1 and W_2 are the weights predicted by our model. In this setting, these weights not only reduce the error of each individual technique, but also implicitly account for the MIS weighting, as a unified learned reweighting scheme. Notably, since the balance heuristic itself is not an optimal choice for MIS weights [Grittmann et al. 2019; Hua et al. 2025; Kondapaneni et al. 2019], our method can further benefit from learning more effective MIS weights.

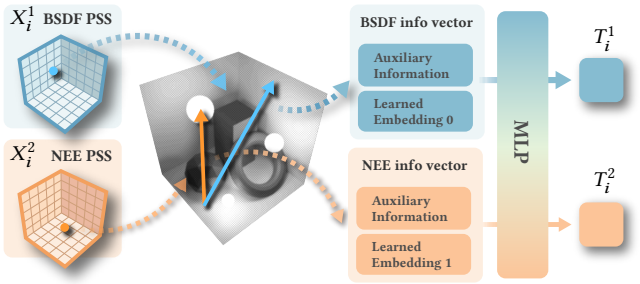


Fig. 18. Direct illumination, Sample and tokenize. In each pass, we draw a 6D random vector, where $X_i^1 \in [0,1]^3$ and $X_i^2 \in [0,1]^3$ serve as PSS samples for BSDF importance sampling and NEE, respectively. The ray tracer then draws a BSDF path and an NEE path accordingly, collecting the corresponding auxiliary information for each. We concatenate these features with different learned embeddings to distinguish the two sampling strategies, which are further fed into a shared MLP to produce two tokens.

Setup specification. As illustrated in Fig. 18, in each pass our neural sampler outputs a 6D random vector $[X_i^1, X_i^2]$, where $X_i^1 \in [0,1]^3$ is the PSS sample used for BSDF sampling: one dimension selects the scattering lobe (for multi-lobe materials such as plastic), and the remaining two dimensions determine the outgoing direction. Similarly, $X_i^2 \in [0,1]^3$ is the PSS sample for next-event estimation (NEE): one dimension selects a light via the light BVH [Conty Estevez and Kulla 2018; Moreau et al. 2022], and the remaining two dimensions sample a point on the spherical light.

Our path tracer then traces the corresponding BSDF and NEE subpaths and collects two feature vectors with the layout specified in Table 5. Each feature vector is concatenated with a distinct learned embedding to distinguish BSDF and NEE samples, and is fed into the same MLP, producing two tokens T_i^1 and T_i^2. For the next pass, the sampler predicts a 1024-dimensional latent conditioned on all previous tokens $T_0^1, T_0^2, \ldots, T_i^1, T_i^2$. This latent is split into two 8^3 logit grids, which define the distributions of $[X_{i+1}^1, X_{i+1}^2]$.

Training. This experiment is trained on a single RTX 5090 GPU. We first train the integrator alone with random samples for 660k iterations (22 hours). We then jointly train the sampler and integrator for an additional 650k iterations (70 hours). During INet-only training, we use a batch size of 4 scenes per iteration. For joint training, we use a batch size of 2 scenes, with 4 rollouts per scene for GRPO. Each scene is rendered at a resolution of 16^2 with random jittering and 16 SPP[4]. Since our method operates per pixel, it can be applied to arbitrary image resolutions at inference time.

Baselines. We compare our method with different combinations of sampling and reconstruction-based integration techniques. Since the trapezoidal rule is defined only for 1D domains, we focus on regression-based Monte Carlo (RMC) [Salaün et al. 2022] using a first-order polynomial. A zero-order model reduces to standard Monte Carlo, while higher-order polynomials perform worse in our tests. For the MC baselines, we use balance heuristic weights for MIS. Although MIS is not discussed in Salaün et al. [2022], we

[4]To reflect the fact that our approach processes a pair of NEE and BSDF samples at a time, we refer to each such NEE+BSDF path pair as a single "sample" in this subsection.

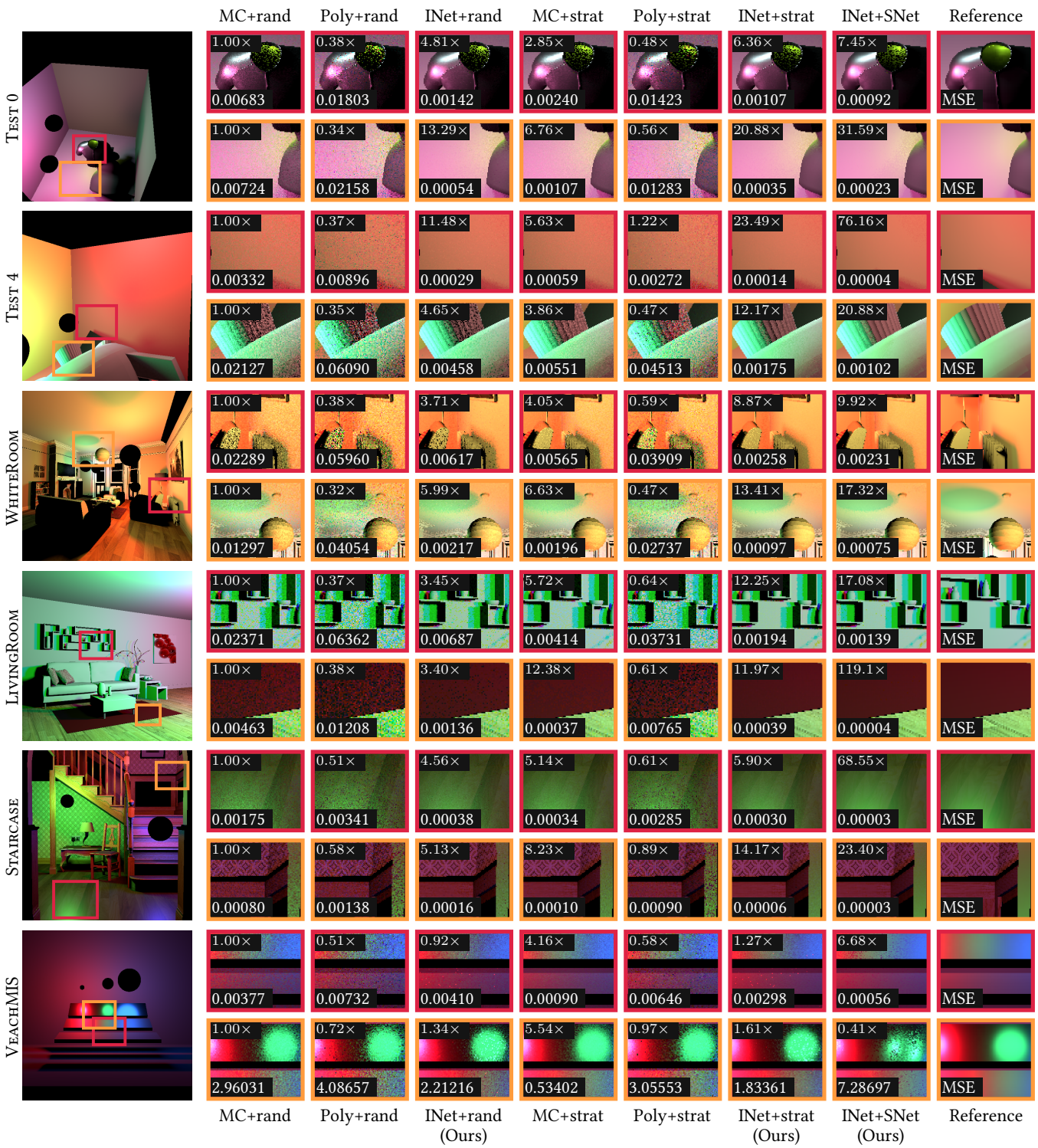


Fig. 19. Direct illumination, Comparison. We compare our method with baseline approaches on both in-distribution test scenes (Test 0 and Test 3) and out-of-distribution scenes (remaining rows), all at 16 SPP. Our approach shows significant improvements in smooth diffuse and glossy regions, shadows and penumbrae, and regions with multi-colored illumination, but struggles with highly specular effects, as in the VeachMIS scene. Since we learn a per-pixel integrator, our method naturally generalizes to high-resolution rendering and textured scenes, preserving high-frequency details even when trained primarily on low-resolution and largely untextured data. The full-image MSE for all cases is reported in Table 3, while we display the MSE for each inset here.

Table 3. Direct illumination, Full-image MSE. We report the MSE over all in- and out-of-distribution scenes at 16 SPP (see Table 10). Our method achieves robust improvements on the in-distribution test cases. For out-of-distribution scenes, the joint variant is slightly worse than INet with stratification on the Teaser scene, and fails on the VeachMIS scene due to its highly specular regions. Nevertheless, our method still demonstrates significant overall improvements.

	Baselines				Ours			
Scene	MC+rand	Poly+rand	MC+strat	Poly+strat	INet+rand	INet+strat	INet+SNet	×
TestScenes (avg)	6.858e-2	2.015e-1	1.908e-2	1.522e-1	7.492e-3	4.140e-3	**2.257e-3**	30.39×
Teaser (Fig. 1)	2.09e-1	5.12e-1	3.70e-2	3.65e-1	3.45e-2	**1.34e-2**	1.38e-2	15.1×
WhiteRoom	7.53e-3	2.12e-2	1.88e-3	1.56e-2	1.81e-3	9.75e-4	**7.13e-4**	10.6×
LivingRoom	2.32e-2	5.93e-2	3.48e-3	3.50e-2	5.83e-3	1.52e-3	**9.53e-4**	24.3×
Staircase	4.44e-3	1.05e-2	1.36e-3	7.69e-3	1.79e-3	8.98e-4	**7.77e-4**	5.72×
VeachMIS	1.68e-1	2.36e-1	**3.00e-2**	1.99e-1	1.35e-1	9.70e-2	2.73e-1	0.614×

also use balance heuristic weights to decompose the problem into NEE and BSDF integrals sperately. We then apply RMC separately to these two 3D integrals, each with a 3D first-order polynomial model. For stratified sampling, we use progressive multi-jittered (PMJ) sequences [Bainbridge 2022; Christensen et al. 2018].

While more advanced techniques (e.g., path guiding and denoising) and their adaptive sampling variants [Firmino et al. 2023; Huo et al. 2020] exist, we do not directly compare to them as they assume different settings (e.g., using cross-pixel or scene-level information). Moreover, our method is mostly complementary and can be combined with them: as shown here, our approach can be applied on top of NEE and BSDF sampling; path guiding can similarly serve as yet another base PSS. Most denoisers operate inter-pixel, whereas ours operates intra-pixel, which makes them naturally compatible.

Results. In Table 3 and Fig. 19, we compare our method with all baselines on both in-distribution test scenes and out-of-distribution scenes. For test scenes generated in the same way as the training set, our joint sampler–integrator achieves 10–60× reduction in MSE compared with basic Monte Carlo with random sampling. The integrator-only variant with stratified samples consistently ranks second and outperforms all other baselines.

On more out of distribution scenes, our method generally still improves over the baselines, with the exception of the VeachMIS scene, where performance degrades in highly specular regions. The potential underlying cause is analyzed further below.

We also provide a rendered video in the supplementary material to demonstrate that our estimates are also temporally coherent.

Learned strategies. We next interpret the learned behavior of our estimator. Figure 20 shows the contributions of BSDF and NEE samples to the final image. In the INet+SNet case, NEE alone produces an almost noise-free estimate of the radiance, so BSDF samples contribute negligibly. This is consistent with the discussion in optimal MIS [Kondapaneni et al. 2019]: when one strategy is nearly optimal, the others should be effectively ignored – something the balance heuristic cannot achieve, but our INet naturally learns to do.

Figure 21 illustrates how SNet selects lights in the LBVH-based primal sample space, compared against uniform PSS baselines. Overall, SNet tends to distribute samples more uniformly across lights. In the second row, the LBVH favors the upper light because it ignores visibility, whereas our method assigns more samples to the

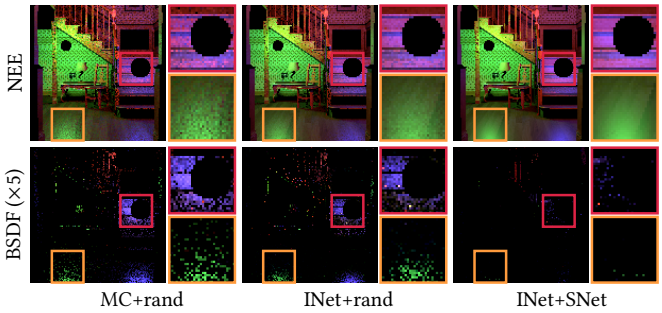


Fig. 20. Direct illumination, MIS Behavior. We visualize the contributions of the NEE and BSDF samples separately, such that the final result is given by the sum of the first and second rows (for clarity, we scale the BSDF contribution by a factor of five). Monte Carlo (MC) relies on balance heuristic weights, and its variance arises from both the NEE and BSDF components. Our INet-only variant achieves a substantially lower-variance NEE component and also reduces the contribution from BSDF samples. With INet+SNet, the NEE contribution becomes almost noise-free, while the BSDF contribution is nearly suppressed. This behavior likely reflects the fact that, in this setting, reducing the variance of BSDF samples is significantly more difficult than for NEE samples, and the end-to-end model therefore tends to downweight or even effectively ignore the BSDF component.

bottom light, which actually contributes to the shading point. In this case, the improvement can be understood through classical adaptive importance sampling and path guiding principles [Figueiredo et al. 2025; Vévoda et al. 2018; Wang et al. 2021]. Interestingly, SNet also benefits from a more uniform allocation even when there is no occlusion (third row), a behavior that would be suboptimal from a traditional importance sampling perspective. This observation echoes Fig. 4 and suggests that our learned INet can benefits from informative samples rather than high-valued ones.

Figure 22 illustrates how SNet samples spherical area lights. Figure 23 further visualizes how it draws BSDF samples, revealing a somewhat opportunistic strategy. As shown in Fig. 20, BSDF samples contribute little to the final estimate in most cases; consequently, SNet learns to concentrate samples in low-contribution regions, where individual contributions are small and variance is low. While this behavior is effective for diffuse and moderately glossy surfaces,

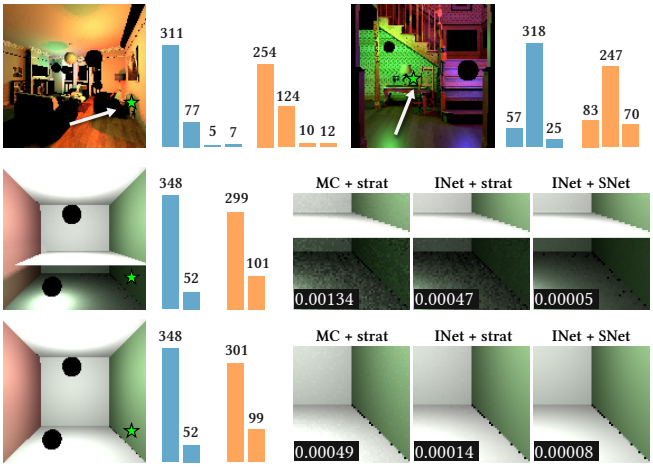


Fig. 21. Direct illumination, Light selection. We visualize how different sampling strategies select lights. Each bar indicates the number of NEE samples, within a 5×5 region around the green star pixels, that are assigned to each area light. The blue bars show how the PMJ sequence selects lights via the light BVH (LBVH), while the orange bars illustrate the behavior of our SNet. Our SNet tends to distribute samples more uniformly across different lights. In the second and third rows, we consider a scene in which the upper light is approximately ten times brighter than the lower one. As a result, LBVH allocates most samples to the upper light, whereas our method consistently assigns more samples to the lower light. **(Second row)** In the presence of an occluder, the naive LBVH strategy performs poorly, whereas our method achieves a significant improvement by allocating more samples to the light that actually contributes, consistent with the intuition of path guiding. **(Third row)** However, even without an occluder, our SNet still favors the lower light and continues to yield a better estimate, in contrast to typical path-guiding behavior, potentially echoing the discussion in Fig. 4.

it becomes detrimental in highly specular cases, where accurate estimation relies critically on BSDF sampling. This explains the poor performance of our method on the VeachMIS scene. We attribute this to a bias in the training data: strongly specular configurations are rare, as they require both a highly specular surface and a light source aligned with its mirror direction. Addressing this issue likely requires deliberately enriching the dataset with such cases.

6.2 Real-valued integral, generalized winding numbers

Generalized winding numbers [Jacobson et al. 2013] describe how many times a query point $\mathbf{p}$ lies inside or outside a geometry. In 2D, they are defined as an integral over the boundary curve C:

$$\mathrm{gwn}(\mathbf{p}) = \frac{1}{2\pi} \oint_C \mathrm{d}\theta = \frac{1}{2\pi} \oint_C \frac{(\mathbf{x}(s) - \mathbf{p})^\perp \cdot \mathrm{d}\mathbf{x}(s)}{\|\mathbf{x}(s) - \mathbf{p}\|^2}, \tag{5}$$

where θ denotes the angle subtended at $\mathbf{p}$, $\mathbf{x}(s)$ parameterizes points along the curve, and $(\cdot)^\perp$ denotes a perpendicular vector. The interesting part is that the integrand can take both positive and negative values, making it a real-valued integral, unlike rendering.

Dataset. To validate our approach on the 2D winding number problem, we construct a synthetic dataset composed of quadratic Bézier curves, as shown in Fig. 38. Each scene consists of six Bézier curves, with all control points sampled uniformly at random.

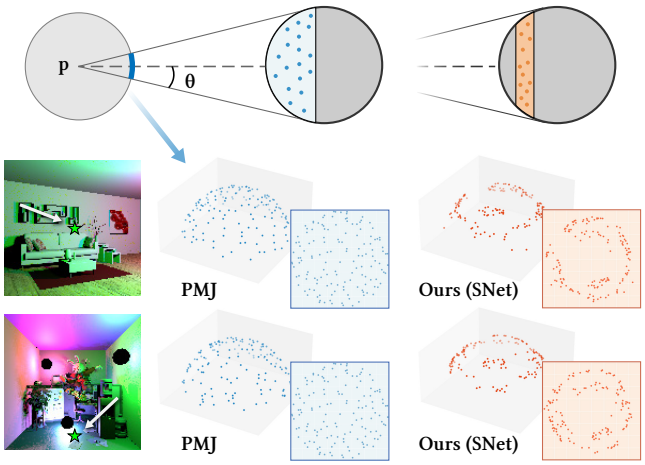


Fig. 22. Direct Illumination, Sphere Sampling. For each shading point $\mathbf{p}$, we use equal–solid-angle sampling to draw NEE samples from the spherical light. We visualize these samples by reprojecting the corresponding subtended angles θ onto the hemisphere. While uniform sampling in the PSS produces a uniform distribution over the spherical cap with respect to $\mathbf{p}$, our SNet learns to concentrate samples along a ring. This increased concentration likely reduces variance, suggesting that the sampler has discovered a favorable bias–variance trade-off for the given dataset. This is intuitive, since boundary samples are usually more informative, while central visibility can often be inferred, especially in our dataset with few thin occluders.

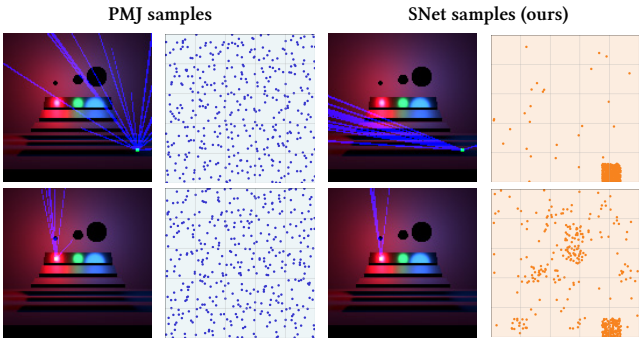
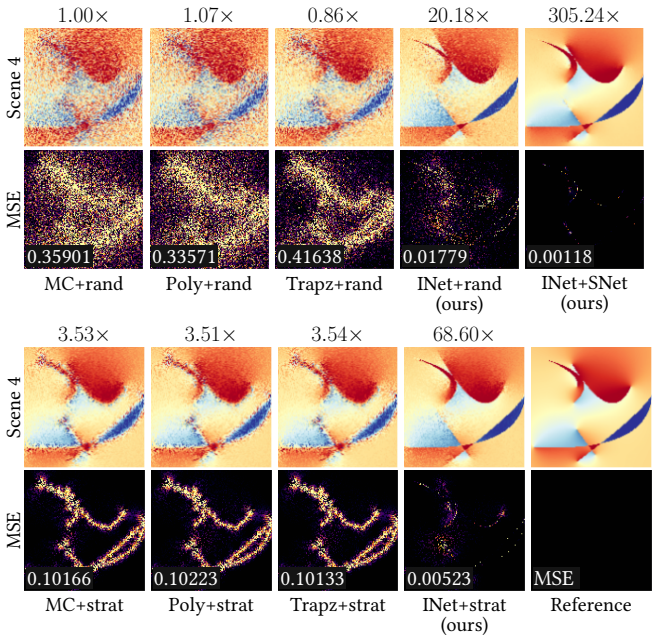


Fig. 23. Direct illumination, BSDF sampling. We visualize how stratified sampling and our SNet draw BSDF samples in diffuse (first row) and specular (second row) regions. In the diffuse case, SNet biases rays toward a low-contribution, visually uninformative region. This behavior is partly explainable: as shown in Fig. 20, BSDF samples are rarely used in the final estimate, and concentrating samples in such low-contribution regions can further reduce variance. However, this strategy breaks down for highly specular surfaces, where NEE alone is insufficient to produce accurate estimates. This explains the poor performance of our method in the VeachMIS scene.

Setup specification. We aim to evaluate the continuous integral in Eq. (5) using numerical integration. We assume that sample points $\mathbf{x}(s)$ can be queried for any $s \in [0, 1]$. For each sample, we provide auxiliary information including the relative position $\mathbf{x}(s) - \mathbf{p}$, the curve normal $\mathbf{n}(s)$, a curve index $I(s) \in \{0, \ldots, 5\}$, and several partial terms of the integrand, such as $\|\mathbf{x}(s) - \mathbf{p}\|^2$ and $1/\|\mathbf{x}(s) - \mathbf{p}\|^2$.



	MC rand	MC strat	Trapz strat	Ours: INet rand	Ours: INet strat	Ours: INet SNet	×
AVG	3.37e-1	3.52e-1	3.54e-1	2.22e-2	5.52e-3	**2.23e-3**	151.1×

Fig. 24. Generalized winding numbers, Results on Test Sets. We report the MSE of all methods using 32 samples per pixel on the test set (see Table 11). The test geometries and boundary conditions are generated in the same manner as the training set, but are not used during training. The best result is shown in **bold** and the second best is underlined. One of the test scene is also visualized in the figure above. Our integrator-only network already significantly outperforms all baseline methods, and the jointly trained estimator can even achieve an 100–300× reduction in MSE.

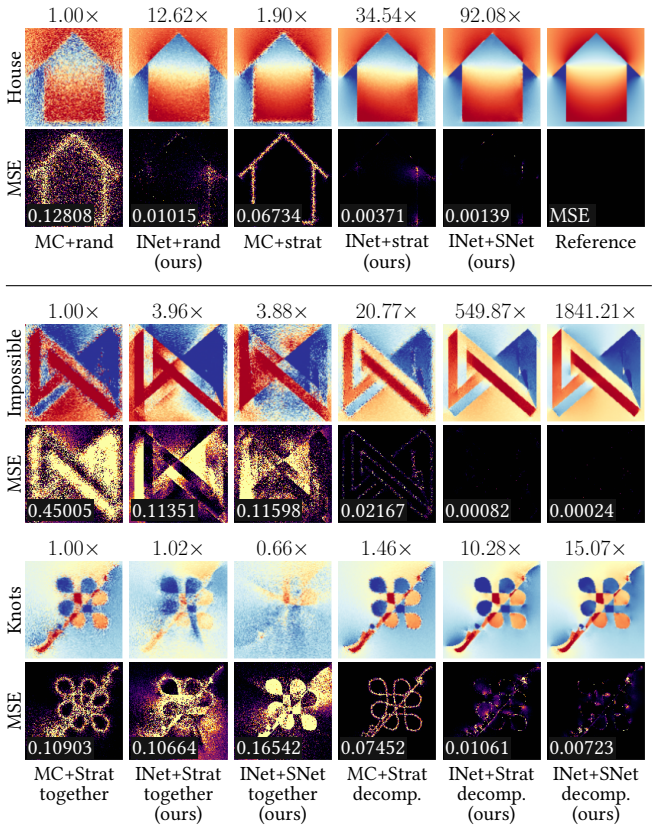


Fig. 25. Generalized winding numbers, Results on OOD Data. We evaluate our method on out-of-distribution (OOD) shapes. House contains 5 line segments, Impossible contains 18 segments, and Knots contains 30 curves. Line segments are represented as Bézier curves via collinear control points, and cases with fewer than six curves are handled by degenerate curves. For shapes with more than six curves, since we assume an index $I_i \in \{0, \ldots, 5\}$ as network input, we consider two strategies: (1) directly reducing the index modulo 6, denoted as *together* (first three cols), and (2) decomposing the shape into multiple six-curve subsets and summing their contributions using linearity, denoted as *decomp.*. For the *decomp.* setting, we use 32 SPP per subproblem. To enable a fair equal-sample comparison, we therefore average K independent 32-SPP estimates for the *together* setting, where $32 \times K$ equals the total number of curves. While the *together* strategy may deviate too much from the training distribution to be beneficial, our method exhibits robust improvements under the *decomp.* strategy.

For reference, we use the analytical solution available for Bézier curves. Note, however, that our estimator is not given the curve representation and does not know that the geometry is Bézier; we also do not provide control points as auxiliary information.

As baselines, because the problem is formulated as a continuous integral, many fast methods designed for discrete representations, such as point clouds or primitive soups [Barill et al. 2018; Dahm and Keller 2017], are not directly applicable. We therefore compare against numerical quadrature rules, as discussed in Section 4.

Results. In Fig. 24, we compare our methods against all 1D quadrature baselines using both random noise and uniform jittered sampling. Both the integrator-only model and the jointly trained estimator achieve significant improvements over all baselines, with particularly large gains in regions close to the geometry. In Fig. 25, we further evaluate the method on scenes that are outside the training distribution, and discuss its behavior on more complex geometries.

In Fig. 26, we visualize the sampling and reweighting strategies learned by our joint model. The sampler generally places more samples near discontinuities, peaks, and valleys of the integrand. The reweighting strategy varies with the query point. When the query point is far from the curve geometry, most samples are downweighted. While this may appear to cause underestimation, remember the integrand in this task includes both positive and negative values. As a result, downweighting negative samples can increase the final estimate and compensate the reduction from positive samples. Notably, our method handles negative values naturally in a unified manner, whereas Monte Carlo methods often require explicit positivization strategies [Owen 2013]. When the query point is close

Table 4. Transmittance, MSE comparison. We compare our method against biased ray marching [Kettunen et al. 2021] (which typically attains lower MSE than its unbiased variant), track-length [Coleman 1968; Novák et al. 2018], ratio tracking [Novák et al. 2014], the trapezoidal rule and regression-based Monte Carlo [Salaün et al. 2022] using degree-3 polynomials on ray marching, as well as the Jackknife estimator [Peters 2025]. For ray-marching-based approaches, including ours, we use one ray with 24 queries per pixel (see Table 12). For tracking-based approaches, we continue sampling until more than 24 queries have been performed in previous tracking. For all algorithms, we adopt the simplest setup with a global majorant of 1 and no importance-sampling structures [Kettunen et al. 2021; Peters 2025], which are difficult to construct for our procedural noise fields. Our approach shows significant and robust improvements over all baselines.

								Ours		
Scene	Ray Marching rand	Track-length	Ratio tracking	Trapz strat	Poly strat	Ray Marching strat	Jackknife strat	INet strat	INet SNet	×
TestScenes (avg)	7.49e-3	1.71e-2	9.38e-3	1.33e-4	1.63e-4	1.50e-4	6.65e-4	1.05e-4	**2.39e-5**	338.4×
Donut	6.67e-3	1.63e-2	8.46e-3	8.21e-5	8.81e-5	8.57e-5	3.79e-4	7.75e-5	**1.30e-5**	515.2×
Letters	5.33e-3	1.10e-2	7.27e-3	5.84e-5	6.49e-5	5.82e-5	2.60e-4	1.85e-4	**3.99e-5**	133.6×

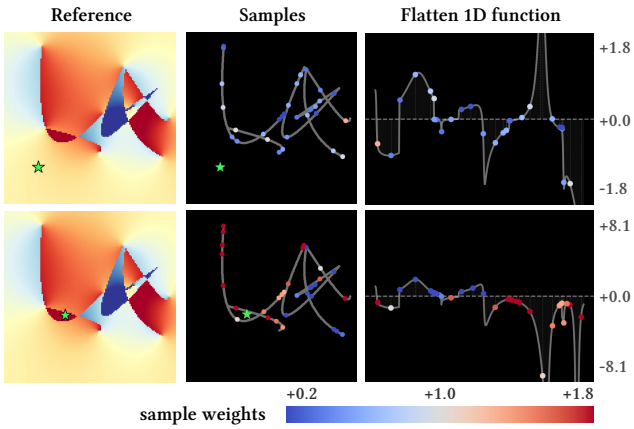


Fig. 26. Generalized winding numbers, Learned strategies. We visualize the learned sampling and reweighting strategies for different query points, marked by green stars. The middle column shows how samples are distributed along the curve, and the right column shows the corresponding 1D integrand $f(s)$. We can observe that the sampler places more samples near discontinuities, peaks, and valleys of the integrand. Weights are shown by color. When the query point is far from the shape and the winding number is close to zero, most samples receive weights below one. In contrast, when the point is close to the geometry and lies in a strongly negative region, the corresponding negative samples are assigned high weights.

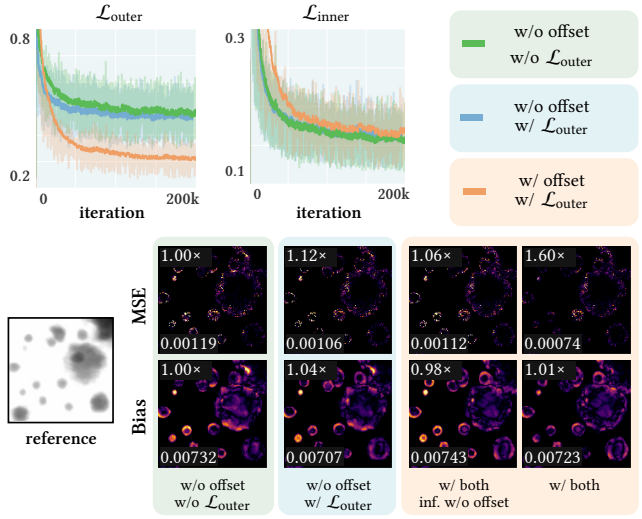


Fig. 27. Transmittance, Design choice ablation. We compare the training loss curves of three integrator designs: (1) without the offset term and trained without $\mathcal{L}_{\text{outer}}$; (2) without the offset term but trained with $\mathcal{L}_{\text{outer}}$; and (3) with the offset term and trained with $\mathcal{L}_{\text{outer}}$. The third, full model achieves the lowest final error, which is also reflected in the MSE comparison in the second row. The third column further reports the same model as in (3) but evaluated at inference time without the offset term. Overall, the full model attains the lowest MSE. However, this does not necessarily imply lower bias, since our training objective only explicitly minimizes MSE. All ablation models here are trained in the case of 12 SPP for 200k iterations.

to the geometry, samples near the curve tend to receive weights larger than one, while samples farther away are downweighted.

6.3 Nonlinearity, transmittance

Another important class of problems involves integrals inside a nonlinear function. A common example is transmittance, defined as

$$T = \exp(-\tau), \quad \tau = \int_0^{t_{\max}} \mu(t)\,\mathrm{d}t \tag{6}$$

where τ is the optical depth, $\mu(t)$ is the volume density at distance t along the ray, and $t_{\max}$ is the distance to the volume boundary. Since non-linear functions do not communicate with expectation in general: $\varphi(\mathbb{E}[X]) \neq \mathbb{E}[\varphi(X)]$, applying a nonlinear function to a Monte Carlo estimate of the inner integral introduces bias, even when the integral itself is unbiasedly estimated.

Dataset. To train the transmittance estimator, we construct a simple synthetic dataset of volumetric densities using a collection of spheres combined with procedural noise, as shown in Fig. 40.

Handling nonlinearity. We introduce two minor modifications to extend our approach to this nonlinear setting. First, in the neural integrator, we let the network predict an additional offset term o_i

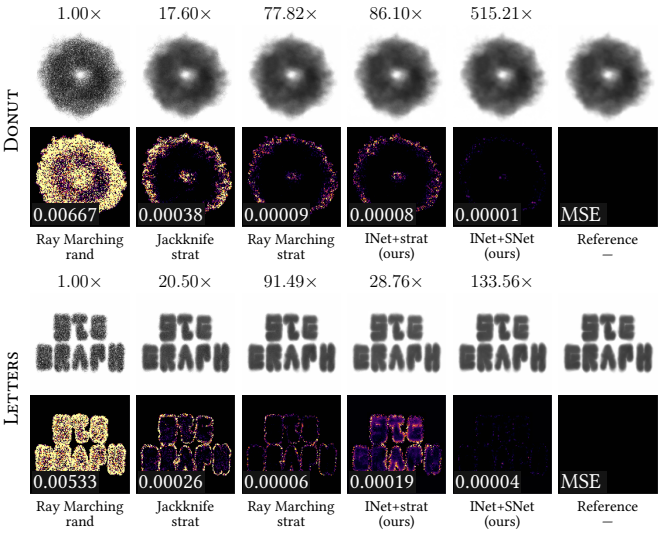


Fig. 28. Transmittance, Results on OOD scenes. We report MSE on two out-of-distribution scenes whose geometry is not composed of spheres, but uses a Donut and a series of Letters. Our joint estimator continues to achieve significant improvements on both in terms of MSE.

along with the weight w_i, and estimate the transmittance as

$$\hat{T} = \exp\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} w_i\, u(X_i)\right)}_{\text{integral estimate}} + \underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} o_i\right)}_{\text{bias compensation}}, \tag{7}$$

where the first term estimates the inner integral as usual, and the offset term is used to compensate for the bias introduced by the exponential nonlinearity. Second, to train the nonlinear integrator in Eq. (7), we use a weighted sum of two loss terms:

$$\mathcal{L} = \lambda_{\text{inner}}\,\mathcal{L}_{\text{inner}} + \lambda_{\text{outer}}\,\mathcal{L}_{\text{outer}}. \tag{8}$$

The inner loss $\mathcal{L}_{\text{inner}}$ measures the mean squared error (MSE) of the estimated inner integral τ, while the outer loss $\mathcal{L}_{\text{outer}}$ measures the MSE of the final transmittance after applying the exponential. The inner loss guides the learning of the weights w_i for estimating the integral, and the outer loss guides the learning of the offset terms o_i to compensate for the bias introduced by the nonlinear exponential. In all experiments, we set $\lambda_{\text{inner}} = 0.2$ and $\lambda_{\text{outer}} = 0.8$. This design, including the offset term and the hybrid loss, is validated in Fig. 27.

Setup specification. We use 24 density queries along each marching ray to estimate the transmittance, treating each query as one sample. As auxiliary information, we include the distance t and decompose the density u into a geometry-induced component u_{geo} and a fractal Brownian motion (fBM) noise component u_{fBM}. In our dataset, all procedural density field is synthesized as

$$u(\mathbf{x}) = \text{clamp}\big(1 - u_{\text{geo}}(\mathbf{x}) + u_{\text{fBM}}(\mathbf{x}),\ 0,\ 1\big)\,. \tag{9}$$

To obtain reference values, we compute the reference optical depth τ_{ref} for $\mathcal{L}_{\text{inner}}$ using 128-query stratified ray marching, and the reference transmittance T_{ref} for $\mathcal{L}_{\text{outer}}$ using 128 samples of ratio tracking. This ensures that both objectives remain unbiased under the ℓ_2 loss [Lehtinen et al. 2018]. Baselines are summarized in Table 4.

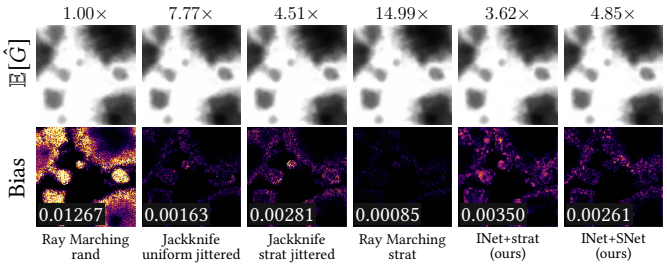


Fig. 29. Transmittance, Bias. We compare the bias of different biased estimators by averaging 256 estimates with 24 queries per pixel. Existing methods show very low bias, likely because the scene is not challenging enough for 24 stratified samples. Our methods do not further reduce the bias, but also exhibit only negligible bias compared to random sampling. In the smoothed distance application, as shown in Fig. 46, our estimator can even significantly reduce the bias compared to the stratified baselines.

Training. This experiment is trained on a single RTX 5090 GPU. The integrator is first trained alone for 640k iterations (10.5 hours), followed by joint training for an additional 64k iterations (6.8 hours). Both stages use a batch size of 4 scenes, and GRPO uses 8 rollouts per scene. All training is conducted with 24 SPP and 32^2 resolution.

Results. We report mean squared error on both in-distribution and out-of-distribution scenes in Table 4, Fig. 28, and Fig. 48. Our approach achieves 100× to 500× MSE reduction compared to standard ray marching with random sampling.

Bias. Unlike in other applications, baseline quadrature rules in this setting also produce biased estimates. In Fig. 29, we compare the bias[5] of different methods at 24 SPP. While our approach does not exhibit lower bias than the stratified baselines, its bias remains comparable. In Appendix E.1, we discuss another application involving a smoothed distance function with a logarithmic nonlinearity instead of an exponential one, where our method achieves even lower bias than the baselines despite being trained with an MSE objective. This difference may stem from the fact that the logarithm is much steeper than the exponential in the relevant regions.

Learned strategies. Our approach learns to draw adaptive, informative samples, with further discussion in Appendix D.

6.4 Recursive integral, Walk on Spheres

The Walk on Spheres (WoS) algorithm [Sawhney et al. 2025] is a grid-free Monte Carlo method for solving linear partial differential equations (PDEs). In this section, we explore how our approach can be applied to the WoS solver for 2D Laplace problems.

Problem specification. In WoS, each *walk* is treated as a sample. Starting from a query point, at step j the algorithm computes the distance to the closest boundary point, denoted by d_j, samples a direction θ_j uniformly, and moves a distance d_j along this direction.

[5] We estimate bias by averaging multiple runs and comparing to the reference, i.e., Bias = $\mathbb{E}[\hat{I}] - I_{\text{ref}}$.

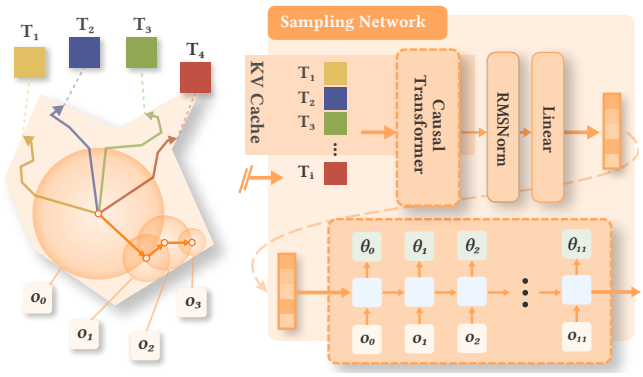


Fig. 30. WoS, Step-level adaptive sampling. In our setting, the sampler must generate a 12-dimensional random vector $\theta_0, \ldots, \theta_{11}$ for each walk. We first encode all previous walks as tokens and use a causal Transformer to produce a latent representation. Instead of directly mapping this latent to a joint distribution over θ, we use it to initialize a recurrent network based on gated recurrent units (GRUs) [Cho et al. 2014]. At each step j, the GRU takes the auxiliary information o_j as input and predicts the distribution of the next direction θ_j. This allows each sample to depend not only on all previous walks but also on the states of earlier steps in the current walk.

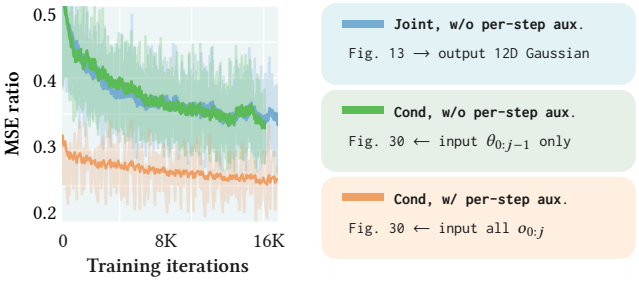


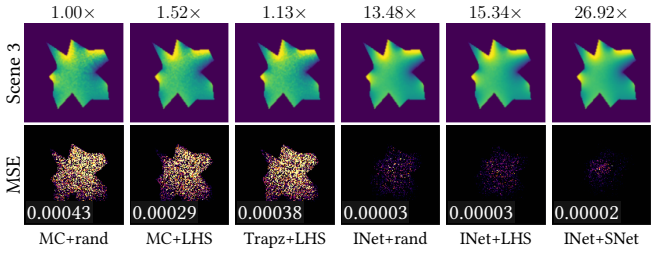
Fig. 31. WoS, Loss curve with different samplers. We compare different sampler network designs by their MSE loss (normalized by the Monte Carlo baseline). The blue curve directly predicts the joint distribution of $\theta_{0:11}$ as a 12D Gaussian, which yields only minor improvement. The green curve uses the RNN in Fig. 30 but takes only the previous direction θ_{j-1} rather than o_j, corresponding to an autoregressive form of Eq. (10). This also brings little gain, indicating that RNN alone is insufficient. The orange curve inputs the full auxiliary information o_j at each step, as shown in Fig. 30, effectively modeling Eq. (11), and achieves a clear performance improvement.

This process is repeated until the distance to the boundary falls below a threshold ϵ. We also set a maximum of 12 steps per walk.

At each step of a walk, we record auxiliary information o_j, including the closest boundary point, its distance, its parametric arc length, its boundary value, and the direction and length relative to the current position and the initial query point. For each walk, its auxiliary information consists of the entire sequence $[o_0, o_1, \ldots, o_{11}]$ of all steps, with missing entries set to zero if the walk ends earlier.

Step-level adaptive sampling. A straightforward way to apply our adaptive sampler in this setting is to model a 12-dimensional joint distribution over all directions θ conditioned on all previous walks:

$$p(\theta_0, \ldots, \theta_{11} \mid T_0, \ldots, T_{i-1}). \tag{10}$$



	MC rand	MC LHS	Trapz LHS	Ours: INet rand	Ours: INet LHS	Ours: INet SNet	×
AVG	2.56e-4	1.70e-4	2.37e-4	2.24e-5	<u>1.86e-5</u>	**9.31e-6**	27.5×

Fig. 32. WoS, Results on Test Sets. We report the MSE of all methods using 32 walks per pixel, averaged over 5 test scenes (see Table 14 for full table). The test geometries and boundary conditions are generated in the same way as the training set, but are not used during training. The best result is shown in **bold** and the second best is <u>underlined</u>. Our approach achieves a significant improvement over all baselines.

However, unlike the previous applications, in WoS we obtain auxiliary information at every step, not only after the entire walk. Therefore, even within a single walk, we can use the information from all previous steps to adaptively choose the next direction. This corresponds to modeling the step directions autoregressively as

$$p(\theta_j \mid T_0, \ldots, T_{i-1}; o_0, \ldots, o_j), \tag{11}$$

where the distribution of θ_j is conditioned not only on all previous walks $T_0, \ldots, T_{i-1}$, but also on the auxiliary information in the earlier steps of the current walk, $o_0, \ldots, o_j$, as implemented in Fig. 30. The benefit of using per-step auxiliary information and modeling Eq. (11) instead of Eq. (10) is confirmed by the ablation study in Fig. 31.

Training. We first train the integrator alone for 276k iterations, which takes 6.2 hours. We then jointly train the integrator and sampler for an additional 80k iterations, which takes 9.8 hours. The integrator-only stage uses a batch size of 8, while the joint training stage uses a batch size of 4 and rolls out 4 trajectories per group. All training uses 32 walks per pixel and 16^2 resolution.

Results. Figure 32 shows the performance of our model on in-distribution test cases, where it achieves a 20–30× improvement over Monte Carlo with the same sample budget. Figure 33 evaluates out-of-distribution (OOD) cases, using highly detailed polygons with complex geometry and high-frequency boundary conditions. Although the improvement is smaller in OOD settings, our method still produces significantly less noise than naive approaches.

As we do not include the segment index as auxiliary input, our model naturally scales to shapes with many more edges, even though it is trained only on star-shaped polygons with 19 edges. We believe that, with a more diverse and carefully designed dataset, the method could be extended to become a more general and robust integrator.

Learned strategies. Figure 34 compares our learned adaptive sampling with baseline methods. Our sampler tends to produce straighter

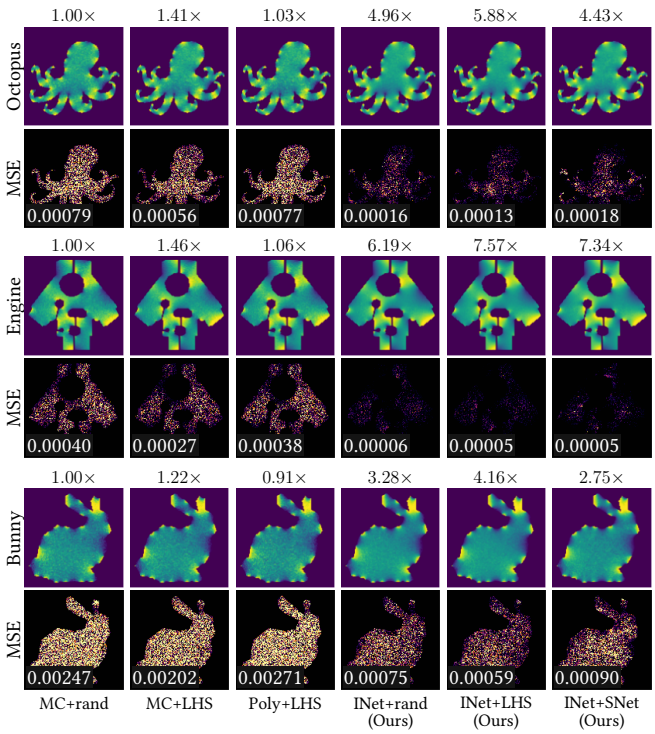


Fig. 33. WoS, Results on OOD Data. We further evaluate our estimator on geometries and boundary conditions that differ significantly from the training data. In particular, we test on highly non–star-shaped domains with interior occlusions and high-frequency boundary values. These shapes have 107, 96, and 286 boundary edges, respectively, whereas the training set contains only star-shaped polygons with 19 edges. Although the improvement ratio decreases, as expected, our method still shows strong advantages.

walks, which leads to a more stratified distribution of walk trajectories and their endpoints. In contrast, primal sample space stratification methods such as Latin hypercube sampling (LHS) only enforce stratification over the random directions at each step. This does not necessarily result in stratified walk paths or endpoints. Our sampler exhibits strong correlations across dimensions. Although this departs from classical notions of uniform stratification, it is intuitive in this setting: correlated steps help explore the boundary more evenly and collect more informative samples for the solver.

In standard WoS for the Laplacian, walk endpoints are distributed according to the Poisson kernel, leading to more samples near the boundary, and all samples are assigned unit weight. In contrast, our sampler prefers more uniform walk trajectories. As a result, the weights must compensate for the Poisson kernel. As shown in Fig. 35, our learned weighting indeed assigns larger weights to nearby endpoints and downweights distant ones. This again highlights the benefit of our joint sampling–integration framework.

Towards nonlinear PDE. In Appendix D, we further demonstrate that our approach extends to nonlinear PDEs (e.g., the p-Laplacian), where the Monte Carlo WoS solvers are not even applicable.

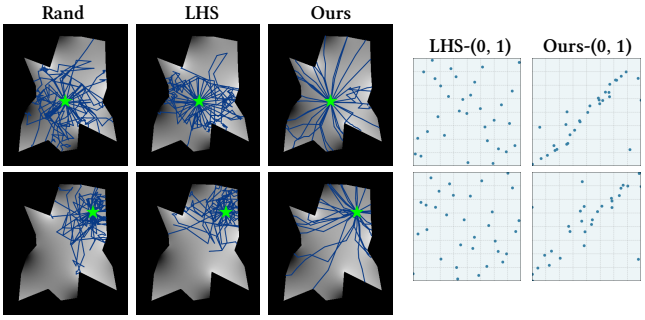


Fig. 34. WoS, Learned sampling strategies. We compare our learned adaptive walks with random sampling and Latin hypercube sampling (LHS). Although LHS enforces stratification over all the sampling directions, this does not guarantee stratification of the resulting walk trajectories. In contrast, our method learns to produce relatively straight walks, which leads to a more stratified distribution of endpoints. We also visualize the projection of the 12D samples onto their first two dimensions. While LHS shows clear stratification in the primal sample space, our samples exhibit strong correlations, which do not follow classical notions of importance sampling or stratification, but nonetheless result in more stratified walk trajectories.

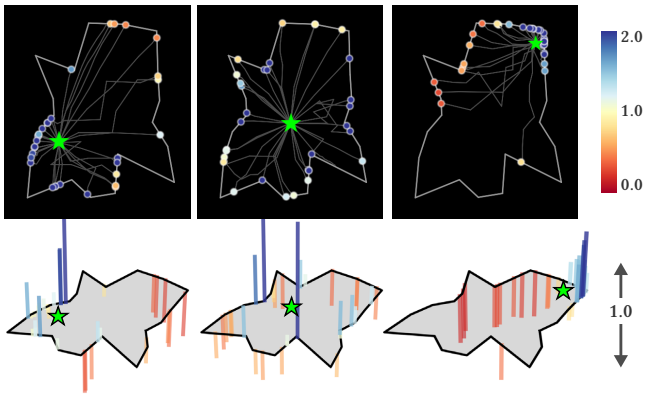


Fig. 35. WoS, Learned weighting strategies. We visualize the learned reweighting factors for different query points. In general, walks that end closer to the query point receive higher weights, while those ending farther away are downweighted. This behavior is intuitive: the endpoints are more uniformly distributed, and our weights take account of the Green's function.

7 DISCUSSION AND FUTURE WORK

Training Objective. Across applications, we use per-pixel MSE to evaluate improvements in individual integrals, distinguishing our method from denoisers that rely on spatial smoothing. Sometimes using SNet may introduce artifacts while reduces MSE, reflecting a bias–variance trade-off under this objective. In practice, more perceptually aligned objectives may be preferred, and our method could also support perceptual losses. Notably, even when trained with MSE, it improves perceptual metrics (Table 13).

Relation to denoising. In this work, we only use samples from the same integral. This design highlights the capability of the integrator itself without any spatial or temporal reuse. A promising

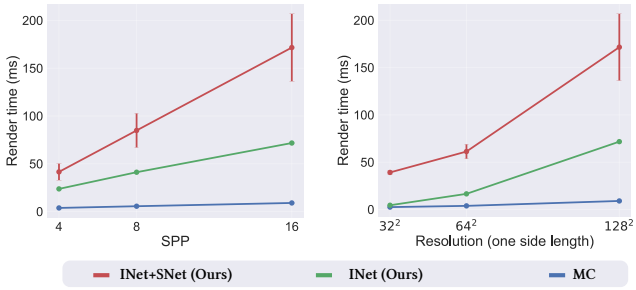


Fig. 36. Time Comparison. We compare the rendering time of our method with standard path tracing for direct illumination. **Left:** time vs. SPP at 128^2 resolution. **Right:** time vs. resolution at 16 SPP. Our full method is approximately 5×–15× slower across settings. Full table in Table 15 to Table 17.

future direction is to jointly model the integral domain and the pixel space [Hachisuka et al. 2008], and to apply attention across samples from different pixels. The adaptive sampler could also be extended to draw joint pixel-integral samples, enabling end-to-end adaptive sampling that accounts for both integration and reconstruction.

Comparing to encoding the whole scene. Zeng et al. [2025a] also address rendering with Transformers by encoding the entire scene as per-triangle tokens and directly predicting the final image, without stochastic sampling or path tracing. However, it is challenging to represent diverse scene attributes—such as textures, volumes, and procedural content—within such a unified encoding, while being scalable to the number of primitives [Xu et al. 2025]. Our approach instead is fully compatible with existing rendering pipelines. We simply replace the final Monte Carlo averaging step with a learned weighted average, without requiring an explicit or complex encoding of the full scene.

Global illumination and advanced path sampling. Since we have discussed how to handle MIS and recursive integrals in Section 6.1 and Section 6.4, respectively, there is no fundamental obstacle to applying our method to global illumination. Another exciting direction is to extend our approach to other light transport algorithms [Georgiev et al. 2012; Hachisuka et al. 2012; Jensen 1996; Lafortune and Willems 1993]. We can also incorporate edge sampling [Li et al. 2018; Soroka et al. 2025] as an additional sampling strategy, as it provides explicit information about discontinuities. On the adaptive sampling side, a promising extension is to allow the sampler to select the sampling strategy for each new sample (e.g., Fig. 20 suggests that BSDF sampling is largely unnecessary for this particular problem), as well as to control Russian roulette and path-splitting decisions [Grittmann et al. 2022; Meyer et al. 2024; Rath et al. 2022].

Efficiency. A major concern of our approach is efficiency. At inference time, the batch size equals the number of pixels, which leads to relatively high computational and memory costs, as shown in Fig. 36. We provide a detailed performance analysis in Appendix F. The $O(N^2)$ time complexity and $O(N)$ memory complexity of the attention mechanism limit the number of samples N that can be used during both training and inference. A promising direction is to investigate whether $O(N)$ attention variants [Gu and Dao 2024; Peng et al. 2023; Sun et al. 2024] can also support our sampling and integration tasks. Another possible direction is to turn our learned behavior into a symbolic algorithm. By analyzing the learned sampler and integrator, we may derive a rule-based method that matches their behavior, either manually or automatically [Bartlett et al. 2025]. If successful, we would no longer need to evaluate a network at inference time, which could further improve the speed.

Memory overhead. Another concern of our method is memory usage: storing 32 tokens (128 floats each) per pixel for a 512^2 image requires about 4 GB, with additional overhead during computation. However, our per-pixel formulation naturally supports tiled processing when memory is limited. Linear attention variants, if applicable, could also further reduce this memory cost to $O(1)$.

Consistency. The convergence of our estimator is driven by MSE minimization, without theoretical guarantees of unbiasedness or consistency, where Monte Carlo methods are particularly strong. A similar issue arises in neural denoising. One possible solution is to combine the learned biased estimate with an unbiased estimator [Gu et al. 2022; Zhou et al. 2025], so that the bias is gradually reduced. Another possibility is to constrain the predicted weights in specific ways to limit bias, as discussed in more detail in Appendix C.2.

Generalization and specialization. While our method shows generalizability to some OOD settings, a key challenge is constructing training datasets that capture the target distribution, just like all other learning-based methods. Notice that some gains arise from specialization to specific integrands (e.g., Fig. 10), but we also aim for broad applicability once trained. Building high-fidelity scene datasets balancing generalization and specialization is an important direction for future work.

8 CONCLUSION

We analyze the limitations of Monte Carlo integration methods and propose an end-to-end framework that jointly learns a sampler and an integrator. This formulation enables the estimator to:

(1) Remove the strict dependence on importance sampling by replacing the Monte Carlo estimator, and enable drawing and using adaptive samples in a more principled manner;
(2) Leverage auxiliary information that is typically ignored by Monte Carlo methods, allowing both the sampler and the integrator to extract more information from each sample;
(3) Enable learning problem-specific priors that specialize the estimator to a given family of problems and integrands.

Our method is broadly applicable across a wide range of tasks within a unified framework, and is inherently compatible with many existing path tracing techniques. We hope it can also benefit other applications in rendering and scientific computing, and motivate further exploration of its combination with established techniques.

ACKNOWLEDGMENTS

This work was funded in part by NSF Grants 2127544, 2212085, 2238839, 2341952 and ONR grant N00014-23-1-2526. We also acknowledge an NVIDIA Fellowship, gifts from Adobe, Google, Activision, and Qualcomm, the Ronald L. Graham Chair and the UC San

Diego Center for Visual Computing. Ramamoorthi acknowledges a part-time appointment at NVIDIA. Scene assets are courtesy of Sketchfab users Giora, crazyshroomz, maxpanysh, and Benedikt Bitterli's Rendering Resources. We also thank Chong Zeng, Bing Xu, and Weijun Dong for helpful discussions.

REFERENCES

Abdalla GM Ahmed and Peter Wonka. 2021. Optimizing dyadic nets. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–17.

Abdalla G. M. Ahmed, Hélène Perrier, David Coeurjolly, Victor Ostromoukhov, Jianwei Guo, Dong-Ming Yan, Hui Huang, and Oliver Deussen. 2016. Low-discrepancy blue noise sampling. *ACM Trans. Graph.* 35, 6, Article 247 (Dec. 2016), 13 pages. https://doi.org/10.1145/2980179.2980218

Josh Bainbridge. 2022. *OpenQMC sampling library for graphics applications.* https://github.com/AcademySoftwareFoundation/openqmc

Steve Bako, Mark Meyer, Tony DeRose, and Pradeep Sen. 2019. Offline deep importance sampling for Monte Carlo path tracing. *Computer Graphics Forum* 38, 7 (2019), 527–542.

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073708

Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Rafał Mantiuk. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 60, 11 pages. https://doi.org/10.1145/3588432.3591562

Sai Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Fredo Durand, Aaron Lefohn, and Yong He. 2023. SLANG.D: Fast, Modular and Differentiable Shader Programming. *ACM Transactions on Graphics (SIGGRAPH Asia)* 42, 6 (December 2023), 1–28. https://doi.org/10.1145/3618353

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020), 245:1–245:18.

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 4 (2018), 43:1–43:12.

Deaglan J Bartlett, Harry Desmond, Pedro G Ferreira, and Gabriel Kronberger. 2025. Introduction to Symbolic Regression in the Physical Sciences. *arXiv preprint arXiv:2512.15920* (2025).

Chuhao Chen, Yuze He, and Tzu-Mao Li. 2024. Temporally Stable Metropolis Light Transport Denoising using Recurrent Transformer Blocks. *ACM Trans. Graph.* 43, 4, Article 123 (July 2024), 14 pages. https://doi.org/10.1145/3658218

In-Young Cho, Yuchi Huo, and Sung-Eui Yoon. 2021. Weakly-supervised contrastive learning in path manifold for Monte Carlo image reconstruction. *ACM Trans. Graph.* 40, 4, Article 38 (July 2021), 14 pages. https://doi.org/10.1145/3450626.3459876

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

Per Christensen, Andrew Kensler, and Charlie Kilpatrick. 2018. Progressive multi-jittered sample sequences. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 21–33.

W. A. Coleman. 1968. Mathematical Verification of a Certain Monte Carlo Sampling Technique and Applications of the Technique to Radiation Transport Problems. *Nuclear Science and Engineering* 32, 1 (1968), 76–81. https://doi.org/10.13182/NSE68-1

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

Miguel Crespo, Adrian Jarabo, and Adolfo Muñoz. 2021. Primary-space Adaptive Control Variates Using Piecewise-polynomial Approximations. *ACM Trans. Graph.* 40, 3, Article 25 (July 2021), 15 pages. https://doi.org/10.1145/3450627

Ken Dahm and Alexander Keller. 2017. Learning light transport the reinforced way. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 73, 2 pages. https://doi.org/10.1145/3084363.3085032

Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2022. Objaverse: A Universe of Annotated 3D Objects. *arXiv preprint arXiv:2212.08051* (2022).

Bastien Doignies, Nicolas Bonneel, David Coeurjolly, Julie Digne, Loïs Paulin, Jean-Claude Iehl, and Victor Ostromoukhov. 2023. Example-Based Sampling with Diffusion Models. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 63, 11 pages. https://doi.org/10.1145/3610548.3618243

Bastien Doignies, David Coeurjolly, Nicolas Bonneel, Julie Digne, Jean-Claude Iehl, and Victor Ostromoukhov. 2024. Differentiable Owen Scrambling. *ACM Trans. Graph.* 43, 6, Article 255 (Nov. 2024), 12 pages. https://doi.org/10.1145/3687764

Fredo Durand. 2011. A frequency analysis of Monte-Carlo and other numerical integration schemes. (2011).

Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X Sillion. 2005. A frequency analysis of light transport. *ACM Trans. Graph. (Proc. SIGGRAPH)* 24, 3 (2005), 1115–1126.

Pedro Figueiredo, Qihao He, Steve Bako, and Nima Khademi Kalantari. 2025. Neural Importance Sampling of Many Lights. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 51, 10 pages. https://doi.org/10.1145/3721238.3730754

Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2023. Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 32, 11 pages. https://doi.org/10.1145/3588432.3591537

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31, 6 (2012), 192:1–192:10.

Zoubin Ghahramani and Carl Rasmussen. 2002. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer (Eds.), Vol. 15. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2002/file/24917db15c4e37e421866448c9ab23d8-Paper.pdf

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Trans. Graph.* 38, 4, Article 125 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322954

Gene H. Golub and John H. Welsch. 1969. Calculation of Gauss Quadrature Rules. *Math. Comp.* 23, 106 (1969), 221–s10. http://www.jstor.org/stable/2004418

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-aware Multiple Importance Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (2019), 152:1–152:9.

Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. Efficiency-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2022)* 41, 4, Article 80 (jul 2022), 12 pages. https://doi.org/10.1145/3528223.3530126

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*.

Jeongmin Gu, Jose A. Iglesias-Guitian, and Bochang Moon. 2022. Neural James-Stein Combiner for Unbiased and Biased Renderings. *ACM Trans. Graph.* 41, 6, Article 262 (Nov. 2022), 14 pages. https://doi.org/10.1145/3550454.3555496

Seymour Haber. 1969. Stochastic quadrature formulas. *Math. Comp.* 23, 108 (1969), 751–764.

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. (Proc. SIGGRAPH)* 27, 3 (2008), 33:1–33:10.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31, 6 (2012), 191:1–191:10.

Yong He, Kayvon Fatahalian, and Theresa Foley. 2018. Slang: language mechanisms for extensible real-time shading systems. *ACM Trans. Graph.* 37, 4, Article 141 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201380

Qingqin Hua, Pascal Grittmann, and Philipp Slusallek. 2025. Correct your balance heuristic: Optimizing balance-style multiple importance sampling weights. *ACM Trans. Graph.* 44, 4, Article 98 (July 2025), 14 pages. https://doi.org/10.1145/3730819

Yuchi Huo, Rui Wang, Ruzahng Zheng, Hualin Xu, Hujun Bao, and Sung-Eui Yoon. 2020. Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning. *ACM Trans. Graph.* 39, 1, Article 6 (Jan. 2020), 17 pages. https://doi.org/10.1145/3368313

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.* 32, 4, Article 33 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461916

Henrik Wann Jensen. 1996. Global Illumination Using Photon Maps. In *Rendering Techniques (Proc. EGWR)*. Eurographics Association, 21–30.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.* 34, 4, Article 122 (July 2015), 12 pages. https://doi.org/10.1145/2766977

Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. 2021. An unbiased ray-marching transmittance estimator. *ACM Trans. Graph.* 40, 4, Article 137 (July 2021), 20 pages. https://doi.org/10.1145/3450626.3459937

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal multiple importance sampling. *ACM Trans. Graph.* 38, 4, Article 37 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323009

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Compugraphics*. 145–153.

Eric P. Lafortune and Yves D. Willems. 1995. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)*. 11–20. https://doi.org/10.1007/978-3-7091-9430-0_2

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. In *International Conference on Machine Learning*.

Thomas Leimkühler, Gurprit Singh, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2019. Deep point correlation design. *ACM Trans. Graph.* 38, 6, Article 226 (Nov. 2019), 17 pages. https://doi.org/10.1145/3355089.3356562

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6 (2018), 222:1–222:11.

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31, 6 (2012), 194:1–194:9.

Zilu Li, Guandao Yang, Qingqing Zhao, Xi Deng, Leonidas Guibas, Bharath Hariharan, and Gordon Wetzstein. 2024. Neural Control Variates with Automatic Integration. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 10, 9 pages. https://doi.org/10.1145/3641519.3657395

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: foundations of ReSTIR. *ACM Trans. Graph.* 41, 4, Article 75 (July 2022), 23 pages. https://doi.org/10.1145/3528223.3530158

Weiheng Lin, Beibei Wang, Jian Yang, Lu Wang, and Ling-Qi Yan. 2021. Path-based Monte Carlo Denoising Using a Three-Scale Neural Network. *Computer Graphics Forum* 40, 1 (2021), 369–381. https://doi.org/10.1111/cgf.14194 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14194

Haolin Lu, Wesley Chang, Trevor Hedstrom, and Tzu-Mao Li. 2024. Real-Time Path Guiding Using Bounding Voxel Sampling. *ACM Trans. Graph.* 43, 4, Article 125 (July 2024), 14 pages. https://doi.org/10.1145/3658203

Haolin Lu, Delio Vicini, Wesley Chang, and Tzu-Mao Li. 2025. Vector-Valued Monte Carlo Integration Using Ratio Control Variates. *ACM Trans. Graph.* 44, 4, Article 100 (July 2025), 16 pages. https://doi.org/10.1145/3731175

Abhishek Madan and David I. W. Levin. 2022. Fast evaluation of smooth distance constraints on co-dimensional geometry. *ACM Trans. Graph.* 41, 4, Article 68 (July 2022), 17 pages. https://doi.org/10.1145/3528223.3530093

D. Meister and T. Harada. 2025. Geometric Integration for Neural Control Variates. *Computer Graphics Forum* 44 (10 2025). https://doi.org/10.1111/cgf.70275

Joshua Meyer, Alexander Rath, Ömercan Yazici, and Philipp Slusallek. 2024. MARS: Multi-sample Allocation through Russian roulette and Splitting. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 20, 10 pages. https://doi.org/10.1145/3680528.3687636

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive polynomial rendering. *ACM Trans. Graph.* 35, 4, Article 40 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925936

P. Moreau, M. Pharr, and P. Clarberg. 2022. Dynamic many-light sampling for real-time ray tracing. In *Proceedings of the Conference on High-Performance Graphics* (Strasbourg, France) *(HPG '19)*. Eurographics Association, Goslar, DEU, 21–26. https://doi.org/10.2312/hpg.20191191

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proceedings of EGSR)* 36, 4 (June 2017), 91–100. https://doi.org/10.1111/cgf.13227

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *arXiv:1808.03856* (2018).

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural Control Variates. *ACM Trans. Graph.* 39, 6, Article 243 (Dec. 2020), 19 pages. https://doi.org/10.1145/3414685.3417804

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time Neural Radiance Caching for Path Tracing. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4 (2021), 36:1–36:16.

Harald Niederreiter. 1992. *Random number generation and quasi-Monte Carlo methods*. SIAM.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37, 2 (May 2018). https://doi.org/10/gd2jqq

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 33, 6 (2014), 179:1–179:11.

A. O'Hagan. 1987. Monte Carlo is Fundamentally Unsound. *Journal of the Royal Statistical Society. Series D (The Statistician)* 36, 2/3 (1987), 247–249. http://www.jstor.org/stable/2348519

Anthony O'Hagan. 1991. Bayes–hermite quadrature. *Journal of statistical planning and inference* 29, 3 (1991), 245–260.

Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl Rasmussen. 2012. Active learning of model evidence using Bayesian quadrature. *Advances in neural information processing systems* 25 (2012).

Katharina Ott, Michael Tiemann, Philipp Hennig, and François-Xavier Briol. 2023. Bayesian numerical integration with neural networks. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence* (Pittsburgh, PA, USA) *(UAI '23)*. JMLR.org, Article 151, 12 pages.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

Ryan S Overbeck, Craig Donner, and Ravi Ramamoorthi. 2009. Adaptive wavelet rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), 140:1–140:12.

Art B Owen. 1997a. Monte Carlo variance of scrambled net quadrature. *SIAM J. Numer. Anal.* 34, 5 (1997), 1884–1910.

Art B Owen. 1997b. Scrambled net variance for integrals of smooth functions. *The Annals of Statistics* 25, 4 (1997), 1541–1562.

Art B Owen. 2013. Monte Carlo theory, methods and examples.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048* (2023).

Hélène Perrier, David Coeurjolly, Feng Xie, Matt Pharr, Pat Hanrahan, and Victor Ostromoukhov. 2018. Sequences with low-discrepancy blue-noise 2-D projections. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 339–353.

Christoph Peters. 2025. Jackknife Transmittance and MIS Weight Estimation. *ACM Trans. Graph.* 44, 6, Article 204 (Dec. 2025), 16 pages. https://doi.org/10.1145/3763273

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. 2006. *Numerical mathematics*. Vol. 37. Springer Science & Business Media.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Philippe Weier, and Philipp Slusallek. 2022. EARS: efficiency-aware russian roulette and splitting. *ACM Trans. Graph.* 41, 4, Article 81 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530168

Christian P Robert, George Casella, and George Casella. 1999. *Monte Carlo statistical methods*. Vol. 2. Springer.

Corentin Salaün, Adrien Gruson, Binh-Son Hua, Toshiya Hachisuka, and Gurprit Singh. 2022. Regression-based Monte Carlo integration. *ACM Trans. Graph.* 41, 4, Article 79 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530095

Farnood Salehi, Marco Manzi, Gerhard Roethlin, Romann Weber, Christopher Schroers, and Marios Papas. 2022. Deep Adaptive Sampling and Reconstruction Using Analytic Distributions. *ACM Trans. Graph.* 41, 6, Article 259 (Nov. 2022), 16 pages. https://doi.org/10.1145/3550454.3555515

Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2025. State of the Art in Grid-Free Monte Carlo Methods for Partial Differential Equations. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses (SIGGRAPH Courses '25)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/3721241.3734001

Antoine Scardigli, Lukas Cavigelli, and Lorenz K. Müller. 2023. RL-based stateful neural adaptive sampling and denoising for real-time path tracing. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2898, 18 pages.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (Los Angeles, California) *(HPG '17)*. Association for Computing Machinery, New York,

NY, USA, Article 2, 12 pages. https://doi.org/10.1145/3105762.3105770

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-Based Error Control for Irradiance Caching. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31, 6 (2012), 193:1–193:10.

Pradeep Sen, Soheil Darabi, and Lei Xiao. 2011. *Compressive Rendering of Multidimensional Scenes.* Springer Berlin Heidelberg, Berlin, Heidelberg, 152–183. https://doi.org/10.1007/978-3-642-24870-2_7

Manu Sethi. 2012. The Schrödinger distance transform (SDT) for point-sets and curves. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (CVPR '12).* IEEE Computer Society, USA, 198–205.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

Mariia Soroka, Christoph Peters, and Steve Marschner. 2025. Quadric-Based Silhouette Sampling for Differentiable Rendering. *ACM Trans. Graph.* 44, 4, Article 81 (July 2025), 20 pages. https://doi.org/10.1145/3731146

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing* 568 (2024), 127063.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. 2024. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620* (2024).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph. D. Dissertation. Stanford University. Advisor(s) Guibas, Leonidas J.

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *SIGGRAPH.* 419–428.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 4 (2018), 125:1–125:12.

Yu-Chen Wang, Yu-Ting Wu, Tzu-Mao Li, and Yung-Yu Chuang. 2021. Learning to cluster for rendering with many lights. *ACM Trans. Graph.* 40, 6, Article 277 (Dec. 2021), 10 pages. https://doi.org/10.1145/3478513.3480561

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. 1988. A Ray Tracing Solution for Diffuse Interreflection. *Comput. Graph. (Proc. SIGGRAPH)* (1988), 85–92.

Liwen Wu, Sai Bi, Zexiang Xu, Hao Tan, Kai Zhang, Fujun Luan, Haolin Lu, and Ravi Ramamoorthi. 2025. Neural BRDF Importance Sampling by Reparameterization. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25).* Association for Computing Machinery, New York, NY, USA, Article 163, 11 pages. https://doi.org/10.1145/3721238.3730679

Bing Xu, Mukund Varma T, Cheng Wang, Tzumao Li, Lifan Wu, Bartlomiej Wronski, Ravi Ramamoorthi, and Marco Salvi. 2025. A Generalizable Light Transport 3D Embedding for Global Illumination. arXiv:2510.18189 [cs.GR] https://arxiv.org/abs/2510.18189

Xiaofeng Xu and Lu Wang. 2025. Adaptive Multiple Control Variates for Many-Light Rendering. In *Eurographics Symposium on Rendering*, Beibei Wang and Alexander Wilkie (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20251184

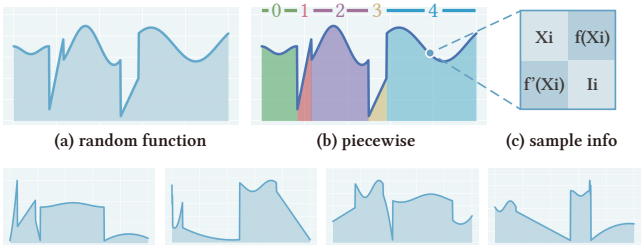
Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. 2018. Deep image-based relighting from optimal sparse samples. *ACM Trans. Graph.* 37, 4, Article 126 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201313

Chong Zeng, Yue Dong, Pieter Peers, Hongzhi Wu, and Xin Tong. 2025a. RenderFormer: Transformer-based Neural Rendering of Triangle Meshes with Global Illumination. In *ACM SIGGRAPH 2025 Conference Papers.*

Zheng Zeng, Markus Kettunen, Chris Wyman, Lifan Wu, Ravi Ramamoorthi, Ling-Qi Yan, and Daqi Lin. 2025b. ReSTIR PG: Path Guiding with Spatiotemporally Resampled Paths. *SIGGRAPH Asia (Conference Track).* https://doi.org/10.1145/3757377.3763813

Chenxi Zhou, Keheng Xu, Mufan Guo, Xianhao Yu, Zhimin Fan, Guihuan Feng, Yanwen Guo, and Jie Guo. 2025. DSCombiner: Double Shrinkage for Combining Biased and Unbiased Monte Carlo Renderings. *ACM Trans. Graph.* 44, 6, Article 203 (Dec. 2025), 12 pages. https://doi.org/10.1145/3763315

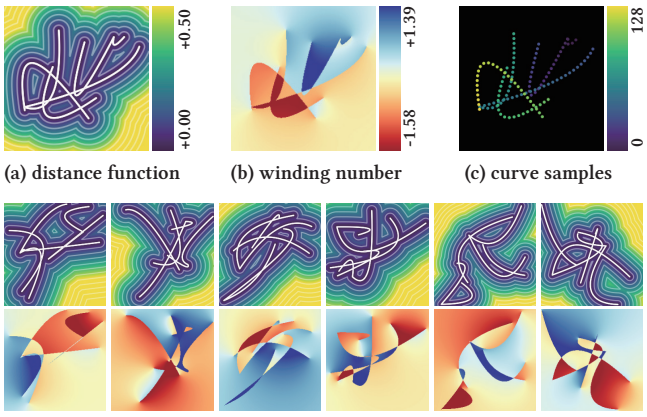
Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Comput. Graph. Forum (Proc. EGSR)* 34, 2 (2015), 667–681.



(a) random function (b) piecewise (c) sample info

(d) more examples in the training dataset

Fig. 37. Toy 1D integration task. **(a)** We consider integrating a family of one-dimensional functions with piecewise smooth structure and discontinuities. **(b)** Each function is constructed by concatenating five smooth segments, randomly chosen from linear, quadratic, and trigonometric functions, with concatenation points placed at random. **(c)** When querying a sample at position X_i, we assume access not only to the function value $f(X_i)$, but also to auxiliary information including the sample position X_i, the function gradient $f'(X_i)$, and the index $I_i \in \{0, \ldots, 4\}$ of the segment in which the sample lies. **(d)** Four additional examples from the training set are shown.



(a) distance function (b) winding number (c) curve samples

(d) examples of random curves in training dataset

Fig. 38. Bézier curves dataset. To validate our approach on 2D geometry problems, we synthesize a dataset in which each sample is composed of six random quadratic Bézier curves, with control points sampled uniformly. Curves 0–1 and 2–3 are connected end to end, while curves 4 and 5 remain isolated, creating shapes with both connected components and disjoint parts. We visualize **(a)** the unsigned distance function, **(b)** the generalized winding number, and **(c)** the mapping from 128 sample $s \in [0, 1]$ to points $\mathbf{x}(s)$ on the curves. **(d)** shows six additional random shapes from the dataset.

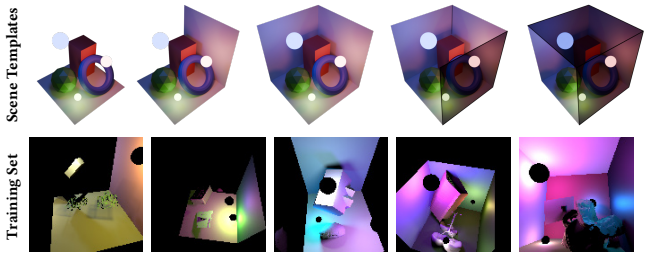


Fig. 39. 3D Scene dataset. To validate our approach on direct illumination, we synthesize a dataset using five scene templates, inspired by Zeng et al. [2025a]. For each scene, we randomly place 1–4 objects from the *Objaverse* dataset [Deitke et al. 2022]. All template quads and objects are assigned random materials (diffuse, conductor, or plastic) with random parameters. Each scene is illuminated by three spherical area lights with random sizes and emission intensities. Cameras are also randomly placed, while ensuring they view the scene contents. We show the five scenes used for training. When showing our scenes and rendering results, we do not visualize the light emission, as it is constant and not part of the direct illumination integral.

Table 5. Direct Illumination, Auxiliary Feature Layout. We illustrate the information collected for each BSDF or NEE subpath. The first three dimensions [0–2] store the primal sample space (PSS) coordinates, and [3–5] store the outgoing direction. These are later encoded using positional encoding of frequency 5 and 3, respectively. Following Lu et al. [2025], we also include per-channel PDF values for light sampling. For simplicity, we assume all lights are spherical area lights and therefore collect light-specific attributes in entries [17–21]. This design can be generalized to other light types by providing different attribute sets together with a light-type embedding, or even by using a separate encoder to produce a type-aware light latent code.

Index	Description
0–2	Input PSS sample X_i^1 (BSDF) or X_i^2 (NEE)
3–5	Sampled outgoing direction ω_i^1 or ω_i^2 in local shading frame
6–8	Incident radiance $f(X_i^1)$ or $f(X_i^2)$ (PDF-divided, w/o MIS)
9	NEE sampling PDF $p_2(X_i^1)$ or $p_2(X_i^2)$ (luminance)
10–12	Ratio of RGB-channel NEE PDF to its luminance PDF
13	BSDF sampling PDF $p_1(X_i^1)$ or $p_1(X_i^2)$ (luminance)
14–16	BSDF value (RGB) with outgoing direction ω_i^1 or ω_i^2
17–19	Normalized direction to the light center (world space)
20	Euclidean distance from the shading point to the light center
21	Radius of the selected (or hit) spherical area light

A APPLICATION SPECIFICATION

A.1 1D toy example

For the 1D toy example, we construct the dataset as described in Fig. 37. Each integrand is formed by concatenating five continuous pieces, each of which is either linear, quadratic, or trigonometric.

A.2 Curve dataset

For the generalized winding number and smooth distance function problems, we construct the curve dataset as described in Fig. 38. Each sample consists of five quadratic Bézier curves, some of which are concatenated, and some are disconnected.

A.3 Direct lighting dataset

For the direct illumination example, we construct the dataset as described in Fig. 39, by randomly placing one to four Objaverse objects into five template scenes. Since the auxiliary information is more involved in this setting, we summarize the full set of auxiliary features provided to the token encoder in Table 5.

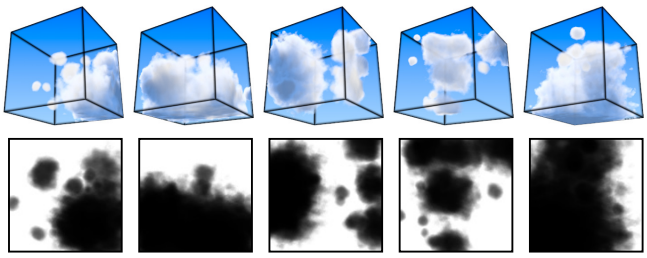


Fig. 40. Procedural volume dataset. For the transmittance task, we generate random volumetric density fields by composing 16 randomly placed and scaled spheres within a $[0, 10]^3$ bounding box. The density is further perturbed using fractal Brownian motion (fBM) noise and then clamped by a global majorant density of 1, resulting in spherical cloud-like structures.

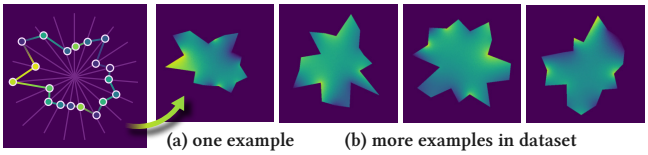


Fig. 41. WoS, Dataset. **(a)** We generate 2D shapes by uniformly dividing the angle into 19 directions and selecting one point along the radius in each direction. A scalar value is assigned to each point. The geometry and Dirichlet boundary values are then obtained by linearly interpolating between neighboring points. **(b)** shows additional examples from the dataset.

A.4 Volumetric dataset

The volumetric dataset is generated as described in Fig. 40, using procedural noise to create cloud-like details and random spheres to define the overall shape.

A.5 2D PDE dataset

For 2D PDE, we train our estimator on a synthetic 2D shape dataset. Each sample is a closed curve generated by radially perturbing a regular nonadecagon at fixed angular directions, resulting in a random star-shaped polygon. A scalar value is assigned to each vertex, and the boundary condition along the curve is defined by linear interpolation between neighboring vertices, as shown in Fig. 41.

B GENERALITY OF SAMPLE REWEIGHTING

We formulate integration as a reweighted average of sample values. This formulation is motivated by the observation that many existing techniques can also be interpreted as sample reweighting.

B.1 Handcrafted quadrature rules as sample reweighting

By definition, quadrature rules approximate an integral F in the form $\hat{F} = \frac{1}{N}\sum_i W_i f(x_i)$, and different rules correspond to particular schemes of sample reweighting. In this section, we explicitly derive the weights used by several 1D rules, as illustrated in Fig. 42.

Rectangle rule as reweighting. The rectangle rule approximates the integrand by a piecewise-constant function and estimates the integral by summing the areas of these bins. We consider a generalized setting where samples $x_i \in [0, 1]$, $i = 1, \ldots, N$, are drawn and sorted such that $x_1 \le \cdots \le x_N$. Using the midpoints between neighboring samples as bin boundaries (see Fig. 42(a)), the estimator can then be written as a weighted average:

$$\hat{F} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{(x_{i+1} - x_{i-1})\, N}{2}}_{W_i} f(x_i), \tag{12}$$

with boundary terms defined as.

$$\begin{cases} x_0 = -x_1 \\ x_{N+1} = 2 - x_N. \end{cases} \tag{13}$$

Trapezoidal rule as reweighting. The trapezoidal rule approximates the integrand with a piecewise-linear function. We extend it to arbitrary sample points, by extending the function as a constant using the left-most and right-most samples at the boundaries, as shown in Fig. 42(b). The resulting estimator is given by:

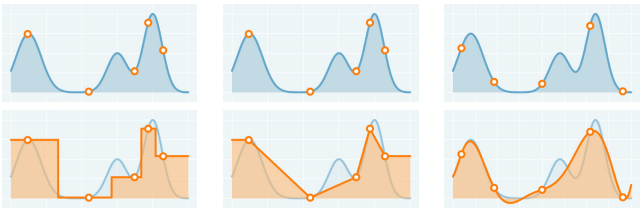


Fig. 42. Discussed Quadrature Rules. **(a)** The rectangle rule reconstructs the integrand using a piecewise-constant approximation. We generalize it to arbitrary sample sequences by placing the discontinuities at the midpoints between neighboring samples. **(b)** We similarly generalize the trapezoidal rule by linearly connecting neighboring samples, resulting in a piecewise-linear approximation. At the boundaries, the function is extended as a constant using the left-most and right-most samples. **(c)** The Gauss–Legendre rule uses predefined sample positions and weights for a given count N.

Table 6. Bayesian quadrature performance. We report the MSE of Bayesian quadrature on our toy 1D problem using random sampling. Even with carefully hand-tuned hyperparameters ($\ell = 0.015$ and $\sigma^2 = 1$), Bayesian quadrature fails to outperform basic Monte Carlo, as the integrands exhibit strong non-smoothness and deviate significantly from the assumed prior.

Case	1	2	3	4	5
MC	**1.15e-2**	**2.33e-2**	**1.48e-2**	**8.55e-3**	**1.54e-2**
BQ	2.22e-1	2.84e-1	7.00e-1	2.39e-1	7.18e-1

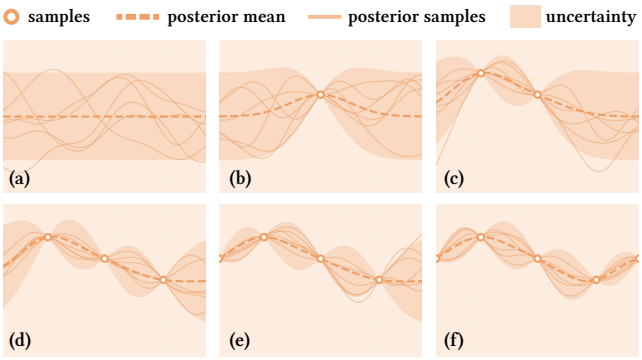


Fig. 43. Adaptive sampling in Bayesian quadrature. **(a)** Bayesian quadrature assumes a Gaussian process prior over integrands. **(b)** Given a sample, it infers a posterior distribution conditioned on the observation, including both a posterior mean and uncertainty. **(c)** Samples are then drawn in regions of high uncertainty to reduce posterior variance. **(d–f)** Subsequent samples are drawn sequentially, conditioned on all previous observations.

$$\hat{F} = \sum_{i=0}^{N} \frac{(x_{i+1} - x_i)}{2} \left[f(x_i) + f(x_{i+1}) \right] \tag{14}$$

$$= \sum_{i=0}^{N} \left(\frac{x_{i+1} - x_i}{2} + \frac{x_i - x_{i-1}}{2} \right) f(x_i) \tag{15}$$

$$= \frac{1}{N} \sum_{i=0}^{N} \underbrace{\left[\frac{N \cdot (x_{i+1} - x_{i-1})}{2} \right]}_{W_i} f(x_i), \tag{16}$$

where we can define the boundary terms as

$$\begin{cases} x_{-1} = x_0 = 0 \\ x_{N+1} = x_N = 1. \end{cases} \tag{17}$$

Gauss–Legendre quadrature as reweighting. Gauss–Legendre quadrature approximates the integrand with a polynomial and evaluates the integral as a weighted average:

$$\hat{F} = \frac{1}{N} \sum_{i=1}^{N} W_i \, f(X_i), \tag{18}$$

where the sample locations X_i and weights W_i are uniquely determined by the quadrature rule for a given N. This rule exactly integrates all polynomials of degree up to $2N-1$ [Golub and Welsch 1969]. Here, each weight W_i depends only on the sample index i and the total number of samples N.

Bayesian quadrature. As illustrated in Fig. 43, Bayesian quadrature assumes a Gaussian process prior over integrands and infers a posterior distribution after each observation, explicitly modeling uncertainty. A natural adaptive sampling strategy is therefore to place new samples in regions of highest uncertainty [Osborne et al. 2012]. In this process, each sample fully exploits information from all previous observations, and no hyperparameters such as N_{init} or N_{adapt} are required. However, Bayesian quadrature relies heavily on the Gaussian process prior assumption, which requires smoothness of the integrand. As a result, it performs poorly on our toy 1D problem with piecewise smooth integrands, as shown in Table 6. However, without Gaussian processes, it becomes difficult to obtain analytical posterior uncertainty and marginalization, which motivates our fully end-to-end learning approach, which avoids explicit probabilistic reconstruction.

Bayesian quadrature [Ghahramani and Rasmussen 2002; O'Hagan 1991] is also a weighted average of the sample values. For a zero-mean Gaussian process prior and a uniform measure $p(x) = 1$ over the domain Ω, the estimator is

$$\hat{F} = \sum_{i=1}^{N} w_i f(X_i) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{(N w_i)}_{W_i} f(X_i), \tag{19}$$

where

$$w = K^{-1} k_p, \tag{20}$$

with

$$K_{ij} = k(x_i, x_j) \tag{21}$$

the kernel (Gram / covariance) matrix of the Gaussian process evaluated at the sample locations, and

$$(k_p)_i = \int_{\Omega} k(x_i, x) \, \mathrm{d}x \tag{22}$$

the kernel mean (or kernel embedding) of the uniform measure.

B.2 Variance reduction methods as sample reweighting

Monte Carlo integration is also a weighted average of sample values, where the simplest form uses a uniform weight of 1. More generally, many variance reduction methods within the Monte Carlo framework can be interpreted as different ways of reweighting samples.

Importance sampling as reweighting. In importance sampling, samples X_i are drawn from a distribution g, and each sample value is reweighted by $1/g(X_i)$:

$$\hat{F} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{g(X_i)} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{1}{g(X_i)}}_{W_i} f(X_i). \tag{23}$$

This reweighting guarantees an unbiased estimator. Variance is reduced when the sampling distribution g is proportional to the magnitude of the integrand $|f|$, so that samples are drawn more often from regions where f has higher absolute value.

Stratification as reweighting. Let $\Omega = \bigsqcup_{s=1}^{S} \Omega_s$ with $A_s = |\Omega_s|$ and draw $X_{s,j} \sim \mathrm{Unif}(\Omega_s)$ for $j = 1, \ldots, N_s$. Then the stratified Monte Carlo estimator of $F = \int_{\Omega} f(x) \, \mathrm{d}x$ is

$$\hat{F} = \sum_{s=1}^{S} \frac{A_s}{N_s} \sum_{j=1}^{N_s} f(X_{s,j}) \tag{24}$$

$$= \sum_{s=1}^{S} \sum_{j=1}^{N_s} \frac{f(X_{s,j})}{N_s \, q_s(X_{s,j})}, \tag{25}$$

where $q_s(x) = \frac{1}{A_s}\mathbf{1}_{x\in\Omega_s}$ is the stratum sampling density. Equivalently, letting $N = \sum_s N_s$ and indexing all samples as X_i with stratum label $s(i)$, we can get the weighted average:

$$\hat{F} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\frac{N\,A_{s(i)}}{N_{s(i)}}}_{W_i} f(X_i). \tag{26}$$

Difference control variates as reweighting. Let h be a control variate with known integral $H = \int h(x)\,\mathrm{d}x$. The classical difference control variate estimator is

$$\hat{F} = H + \frac{1}{N}\sum_{i=1}^{N}\big(f(X_i) - h(X_i)\big). \tag{27}$$

It can be rewritten as a weighted average of the original samples:

$$\hat{F} = \frac{1}{N}\sum_{i=1}^{N}\left[f(X_i) + H - h(X_i)\right] \tag{28}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \underbrace{\left(1 + \frac{H - h(X_i)}{f(X_i)}\right)}_{W_i} f(X_i). \tag{29}$$

Ratio control variates as reweighting. Again, let h be a control variate with known integral $H = \int h(x)\,\mathrm{d}x$. Ratio control variates [Lu et al. 2025] also has a weighted average formulation

$$\hat{F} = \frac{\sum_{i=1}^{N} f(X_i)}{\sum_{i=1}^{N} h(X_i)} \times H \tag{30}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \underbrace{\left(\frac{H}{\frac{1}{N}\sum_{i=1}^{N} h(X_i)}\right)}_{W_i} f(X_i). \tag{31}$$

Multiple importance sampling as reweighting. Let $g_1, \ldots, g_M$ be proposal densities, and draw N_m samples $X_{m,i} \sim g_m$ for $i = 1, \ldots, N_m$, with $\sum_m N_m = N$. The MIS estimator with balance weights is

$$\hat{F} = \sum_{m=1}^{M}\frac{1}{N_m}\sum_{i=1}^{N_m} w_m(X_{m,i})\frac{f(X_{m,i})}{g_m(X_{m,i})}, \tag{32}$$

$$w_m(x) = \frac{N_m g_m(x)}{\sum_{k=1}^{M} N_k g_k(x)}. \tag{33}$$

Equivalently, it can be written as a single weighted average

$$\hat{F} = \frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{N_m} \underbrace{\frac{N\,w_m(X_{m,i})}{g_m(X_{m,i})}}_{W_{m,i}} f(X_{m,i}) \tag{34}$$

$$= \frac{1}{N}\sum_{m,i} \underbrace{\frac{N}{\sum_{k=1}^{M} N_k g_k(X_{m,i})}}_{W_{m,i}} f(X_{m,i}). \tag{35}$$

C ANALYSIS OF THE REWEIGHTING FORMULATION

C.1 Optimality of reweighting

Ideally, the weights would yield a zero-bias, zero-variance estimator:

$$\frac{1}{N}\sum_{i=1}^{N} W_i f(X_i) = F. \tag{36}$$

A trivial but optimal choice is

$$W_i^\star = \frac{F}{f(X_i)} \tag{37}$$

which would recover the exact integral F even with $N = 1$. However, this requires knowing F itself in advance, making it as difficult as the original integration problem.

Notably, many classical variance reduction techniques can be interpreted as approximations of this $W_i^\star$ weight. For importance sampling in Eq. (23), assuming a nonnegative integrand, the optimal sampling density is $g(x) = \frac{f(x)}{F}$, which leads to the weight:

$$W_i = \frac{1}{g(X_i)} = \frac{F}{f(X_i)}. \tag{38}$$

A similar observation holds for difference control variates. The optimal control variate has the form $h(x) = f(x)+C$ with a constant C. The corresponding optimal reweighting becomes

$$W_i = 1 + \frac{H - h(X_i)}{f(X_i)} = 1 + \frac{F + C - (f(X_i) + C)}{f(X_i)} = \frac{F}{f(X_i)}, \tag{39}$$

which again matches the weight $W_i^\star$.

This ideal weight $W_i^\star$ reweights each sample value toward the mean F. This explains the behavior of regression-based Monte Carlo [Salaün et al. 2022] shown in Fig. 11, where the reweighted samples are pulled toward a common central value. This effect arises because it is essentially a difference control variates approach.

However, our approach in Fig. 11 do not follow such pattern, but still achieves much lower variance and MSE than regression-based Monte Carlo, as reported in Table 1. To explain why, notice to let the weighted average match the ground truth integral, Eq. (36) is a single linear constraint on N unknowns, meaning there are infinitely many distinct weight vectors W in the solution space. Monte Carlo variance reduction methods typically determine weights solely as functions of the individual sample values $f(X_i)$, making $W_i^\star = F/f(X_i)$ a natural solution. However, when weights are determined jointly across all samples and information, the space of valid solutions is far richer. For example, a contrived case is to set

$$w_1 = \frac{NF}{f(X_1)}, \qquad w_i = 0 \ \text{ for all } i \neq 1, \tag{40}$$

which also satisfies Eq. (36) and yields an zero-bias, zero-variance estimator. This observation highlights that the feasible solution space of our learned weights is substantially larger than that explored by traditional variance reduction frameworks. By jointly reasoning over all samples and auxiliary information, our approach can navigate this enlarged solution space in a more flexible and effective manner. On the other hand, many classical variance reduction techniques also lie within the solution space of our framework.

C.2 Bias and consistency of reweighting

We now analyze the bias of the reweighting estimator. Since the weights W_i are predicted from all samples $X_{1:N}$, with a joint distribution $p(X_{1:N})$, the expectation should be written as

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} W_i(X_{1:N})\, f(X_i)\right] \tag{41}$$

$$=\int \left[\frac{1}{N}\sum_{i=1}^{N} W_i(X_{1:N})\, f(X_i)\right] p(X_{1:N})\, dX_{1:N} \tag{42}$$

$$=\frac{1}{N}\sum_{i=1}^{N}\int W_i(X_{1:N})\, f(X_i)\, p(X_{1:N})\, dX_{1:N} \tag{43}$$

$$=\frac{1}{N}\sum_{i=1}^{N}\int \left[\int W_i(X_i, X_{\neg i})\, p(X_i, X_{\neg i})\, dX_{\neg i}\right] f(X_i)\, dX_i, \tag{44}$$

where $X_{\neg i} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_N)$. Although this formulation is very general, it is difficult to derive meaningful analysis on the bias. In fact, even the estimator in Eq. (40) can be unbiased under this formulation.

If we further assume that $X_1, \ldots, X_N$ are i.i.d. samples from a density $p(x)$, and that the estimator is permutation-invariant with respect to the samples (which holds when we do not use adaptive sampling and do not apply RoPE [Su et al. 2024]), then:

$$\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} W_i(X_{1:N})\, f(X_i)\right] \tag{45}$$

$$=\frac{1}{N}\sum_{i=1}^{N}\int f(x)\, p(x)\, \mathbb{E}_{X_{\neg i}}[W_i(x, X_{\neg i})]\; dx \tag{46}$$

$$=\int f(x)\, p(x)\, \underbrace{\mathbb{E}[W_1(x, X_{2:N})]}_{\alpha(x)}\; dx, \tag{47}$$

where $\alpha(x)$ denotes the expected weight conditioned on a sample value x, after marginalizing over all possible configurations of $X_{2:N}$.

Under this assumption, the unbiasedness condition becomes

$$\int f(x)\, \alpha(x)\, p(x)\, \mathrm{d}x = \int f(x)\, \mathrm{d}x, \tag{48}$$

$$\int f(x)\, \big[\alpha(x)\, p(x) - 1\big]\, \mathrm{d}x = 0. \tag{49}$$

Interestingly, Unbiased Contribution Weights (UCW) [Lin et al. 2022] can be seen as a special case of Eq. (49). In UCW, the expected weight W satisfies $\alpha(x) = \frac{1}{p(x)}$, which exactly cancels the sampling density $p(x)$ in expectation.

Enforcing the condition in Eq. (49) requires strong structural assumptions. For example, in difference control variates (Eq. (29)), this condition is satisfied by construction, since $\alpha(x)\, p(x) = 1$ holds due to the availability of an analytic integral of the control variate. In our general setting, however, even the marginal function $\alpha(x)$ is not available in closed form. It would be an interesting direction for future work to explore network architectures or constraints that encourage this unbiasedness condition to hold.

When considering consistency, more practical options are available. One approach is inspired by ratio control variates, whose weights, as shown in Eq. (31), gradually converge to 1 as N increases. Following this idea, we can guarantee consistency by blending the predicted weights toward 1 as $N \to \infty$. Another option is to maintain an additional unbiased or consistent estimator and gradually blend our prediction with it [Gu et al. 2022; Zhou et al. 2025], so that the result fully falls back to the consistent estimator as $N \to \infty$.

Table 7. MSE and Variance of 1D Rules. We report the mean squared error (MSE) and variance of 1D quadrature rules with $N = 32$ samples on five random test integrands, together with the improvement ratio of our method over standard Monte Carlo. Poly-3 and Poly-5 denote regression Monte Carlo with degree-3 and degree-5 polynomials, respectively, while Trapz denotes the trapezoidal rule. Our integrator-only approach achieves orders-of-magnitude improvements in MSE compared to basic MC.

Case	MC	Trapz	Poly-3	Poly-5	Ours	ratio
1	1.15e-2	2.40e-3	1.19e-2	1.60e-2	**3.41e-4**	33.9×
	1.15e-2	2.40e-3	1.19e-2	1.55e-2	**2.23e-4**	51.7×
2	2.33e-2	7.03e-3	1.28e-2	5.55e-2	**1.14e-3**	20.4×
	2.33e-2	6.97e-3	1.23e-2	5.46e-2	**0.97e-3**	23.9×
3	1.48e-2	5.11e-3	1.98e-2	5.94e-2	**1.12e-3**	13.2×
	1.48e-2	5.10e-3	1.95e-2	5.57e-2	**0.71e-3**	20.8×
4	8.55e-3	3.08e-3	9.38e-3	1.85e-2	**4.15e-4**	20.6×
	8.55e-3	3.01e-3	9.15e-3	1.72e-2	**3.95e-4**	21.5×
5	1.54e-2	2.29e-3	5.20e-3	4.59e-3	**1.83e-4**	84.4×
	1.54e-2	2.29e-3	5.07e-3	4.51e-3	**1.70e-4**	90.8×

Table 8. Ablation with randomized inputs. We ablate auxiliary information by replacing the sample position X_i with a uniform random value in $[0, 1]$ (Random X_i), the gradient $f'(x_i)$ with Gaussian noise (Random f'), and the piece index I_i with a random integer in $[0, 4]$ (Random I_i). Replacing real inputs with random values significantly degrades performance, indicating that the integrator effectively leverages this information.

Case	MC	Random X_i	Random f'	Random I_i	Full
1	1.15e-2	1.08e-2	9.76e-3	1.02e-2	**3.41e-4**
2	2.33e-2	5.41e-2	9.69e-3	2.54e-2	**1.14e-3**
3	1.48e-2	1.52e-2	1.35e-2	1.48e-2	**1.12e-3**
4	8.55e-3	1.72e-2	2.36e-3	8.77e-3	**4.15e-4**
5	1.54e-2	6.70e-2	3.79e-3	8.11e-3	**1.83e-4**

D ADDITIONAL RESULTS

D.1 Volumetric rendering

Learned strategies. Fig. 49 illustrates the learned sampling and weighting strategies. The adaptive sampler concentrates on informative regions but also does not strictly follow classical importance sampling. For instance, in the first row, regions with density equal to one have large function values, yet are sampled sparsely. Since these regions form a plateau, allocating fewer samples is sufficient, and the reduced sampling density is compensated by weights larger than one. This again confirms the power of adaptive sampling.

Table 9. MSE of 1D Rules and Samplers. We report the mean squared error (MSE) of one-dimensional quadrature rules with $N = 32$ samples using uniform random sampling, stratified sampling, and our joint adaptive sampling. The jointly trained sampler and integrator achieve up to three orders of magnitude lower MSE compared to basic Monte Carlo.

					Ours	
	MC	Trapz	Poly-3	Poly-5	INet	SNet+INet
11	2.28e-2	2.93e-3	7.24e-3	1.03e-2	7.26e-4	**1.2e-5**
	8.87e-4	6.77e-4	7.86e-4	8.23e-4	2.74e-4	1899×
12	1.14e-2	2.43e-3	1.17e-2	9.82e-3	3.58e-4	**6.0e-6**
	6.94e-4	5.04e-4	7.03e-4	7.45e-4	9.10e-5	1908×
13	2.33e-2	7.22e-3	1.34e-2	2.95e-2	1.17e-3	**5.0e-6**
	2.56e-3	2.13e-3	2.46e-3	2.44e-3	5.76e-4	4664×
14	1.57e-2	4.85e-3	2.50e-2	1.04e-1	1.13e-3	**1.4e-5**
	1.26e-3	1.10e-3	1.26e-3	1.30e-3	3.37e-4	1960×
15	7.87e-3	2.55e-3	8.90e-3	2.02e-2	3.76e-4	**4.0e-6**
	6.04e-4	3.51e-4	6.58e-4	4.83e-4	9.10e-5	1968×

E ADDITIONAL APPLICATIONS

E.1 Smoothed distance

Here, we consider another task on the curve dataset: computing a smoothed unsigned distance to the curve [Madan and Levin 2022; Sethi 2012]. We define the smoothed distance at a query point $\mathbf{p}$ as

$$\mathrm{sd}(\mathbf{p}) = -\frac{1}{\alpha}\log\left(\oint_C \exp(-\alpha\|\mathbf{x}(s)-\mathbf{p}\|)\ \mathrm{d}\mathbf{x}(s)\right). \tag{50}$$

Here, the log-integral-exp formulation acts as a soft minimum over distances along the curve, and parameter α controls the smoothness of the approximation. Larger values of α yield results closer to the true unsigned distance, while smaller values produce a smoother field. For simplicity, we use $\alpha = 35$ in all the experiments.

A key challenge of this formulation is the nonlinear concave log applied outside the integral. By Jensen's inequality, we know

$$\log \mathbb{E}[F] \geq \mathbb{E}[\log F], \tag{51}$$

which implies that applying the logarithm after any quadrature estimate introduces bias, even when the integral itself is unbiased.

Our solution. Although our estimator is already biased, a non-zero-variance estimate of the inner integral introduces additional bias after the nonlinear transformation. To address this issue, we modify the integrator network to also predict an offset term o_i that compensates for this bias. The final estimate is then computed as

$$\hat{G} = \mathcal{G}\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} w_i\, f(X_i)\right)}_{\text{integral estimate}} + \underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} o_i\right)}_{\text{bias compensation}}, \tag{52}$$

where $\mathcal{G}$ denotes the nonlinear transformation – in this application, the logarithm – and G is the final quantity to be estimated, namely the smoothed distance here.

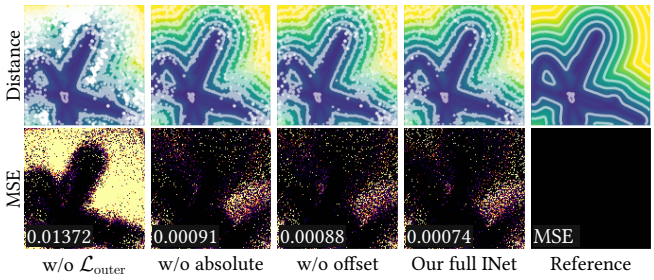


Fig. 44. Smoothed Distance, Ablation. We study how different design choices affect the results of integrator training. Without the outer loss $\mathcal{L}_{\text{outer}}$, regions far from the geometry degrade significantly. In these regions, the inner integral has very low error, but the logarithmic transformation amplifies the error. Therefore, a low training loss can still lead to large errors at inference time. Removing the absolute value or the offset term also degrades performance, leading to slightly worse results than the full model.

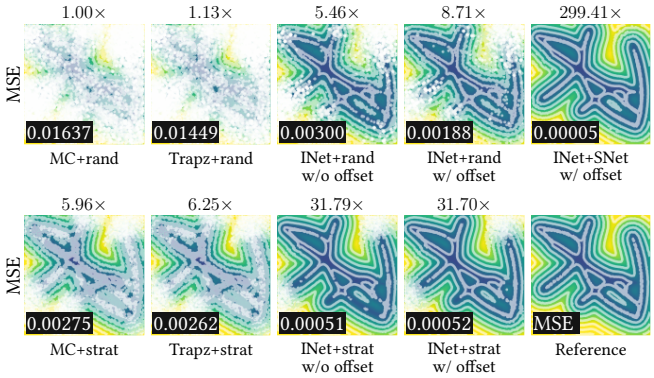


Fig. 45. Smoothed Distance, Comparison. We report the mean squared error (MSE) with 32 samples per pixel (SPP). We train the integrator with and without the offset term separately, with the same Eq. (53) loss, and the version with the offset often achieves lower MSE. Our full model, combined with the adaptive sampler, further outperforms all baseline methods.

To train the integrator, we define the training objective as a weighted combination of two losses:

$$\mathcal{L} = \lambda_{\text{inner}}\,\mathcal{L}_{\text{inner}} + \lambda_{\text{outer}}\,\mathcal{L}_{\text{outer}}, \tag{53}$$

where the inner loss $\mathcal{L}_{\text{inner}}$ is the mean squared error (MSE) of the integral value F itself, and the outer loss $\mathcal{L}_{\text{outer}}$ is the MSE of the final transformed estimate G. Specifically,

$$\mathcal{L}_{\text{inner}} = \mathrm{MSE}(\hat{F}, F_{\text{ref}}), \tag{54}$$

$$\mathcal{L}_{\text{outer}} = \mathrm{MSE}(\hat{G}, G_{\text{ref}}), \tag{55}$$

where the inner loss corresponds to the conventional objective used to optimize the weights w_i, while the outer loss optimizes the offset o_i to reduce the bias of the inner estimate. In all experiments, we set $\lambda_{\text{inner}} = 0.2$ and $\lambda_{\text{outer}} = 0.8$.

Although the non-linear function $\mathcal{G} = \log$ appears in Eq. (50), we use a modified form in practice to improve training stability:

$$\mathcal{G} = \log\Big(\mathrm{clamp}(|\hat{F}|, \epsilon)\Big), \tag{56}$$

Table 10. Direct illumination, Full-image MSE. We report the MSE over all in- and out-of-distribution scenes at 16 SPP. Our method achieves robust improvements on the in-distribution test cases. For out-of-distribution scenes, the joint variant is slightly worse than INet with stratification on the Teaser scene, and fails on the VeachMIS scene due to its highly specular regions. Nevertheless, our method still demonstrates significant overall improvements.

	Baselines				Ours			
Scene	MC+rand	Poly+rand	MC+strat	Poly+strat	INet+rand	INet+strat	INet+SNet	×
TestScene 0	2.49e-2	7.22e-2	5.38e-3	5.34e-2	2.86e-3	1.95e-3	**1.37e-3**	18.2×
TestScene 1	2.80e-2	6.75e-2	8.87e-3	4.66e-2	2.83e-3	1.29e-3	**9.61e-4**	29.2×
TestScene 2	4.90e-3	1.45e-2	2.62e-3	1.25e-2	2.17e-3	1.43e-3	**3.94e-4**	12.4×
TestScene 3	5.70e-2	1.57e-1	1.67e-2	1.30e-1	1.04e-2	6.56e-3	**4.91e-3**	11.6×
TestScene 4	2.29e-1	6.97e-1	5.10e-2	5.18e-1	1.92e-2	9.47e-3	**3.65e-3**	62.9×
Teaser (Fig. 1)	2.09e-1	5.12e-1	3.70e-2	3.65e-1	3.45e-2	**1.34e-2**	1.38e-2	15.1×
WhiteRoom	7.53e-3	2.12e-2	1.88e-3	1.56e-2	1.81e-3	9.75e-4	**7.13e-4**	10.6×
LivingRoom	2.32e-2	5.93e-2	3.48e-3	3.50e-2	5.83e-3	1.52e-3	**9.53e-4**	24.3×
Staircase	4.44e-3	1.05e-2	1.36e-3	7.69e-3	1.79e-3	8.98e-4	**7.77e-4**	5.72×
VeachMIS	1.68e-1	2.36e-1	**3.00e-2**	1.99e-1	1.35e-1	9.70e-2	2.73e-1	0.614×

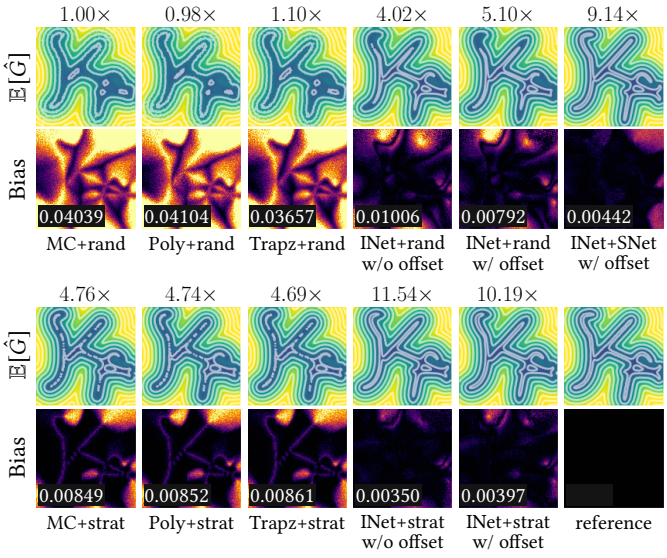


Fig. 46. Smoothed distance, Bias. We also evaluate the bias of all methods by averaging 256 independent estimates with 32 samples per pixel (SPP) to approximate the expected value $\mathbb{E}[\hat{G}]$. Although our models are inherently biased, they achieve significantly lower bias than methods that rely on unbiased estimators for the inner integral, thanks to their lower MSE. In addition, the offset term often helps to further reduce the bias.

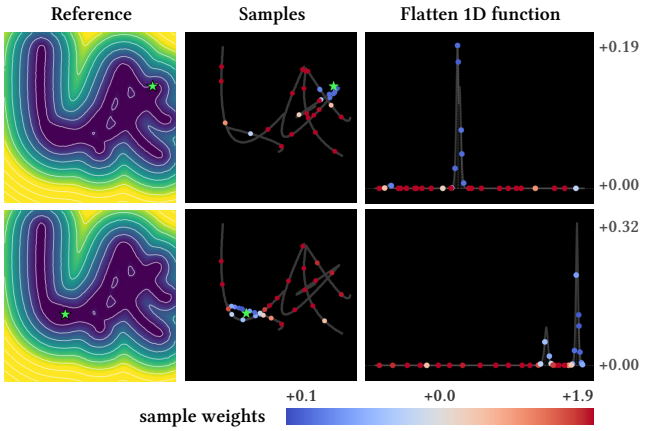


Fig. 47. Smoothed distance, Learned strategies. We visualize the learned sampling and weighting strategies. The query point is shown as a green star. More samples are drawn near the query point, as seen by the concentration of blue points around the star in the middle column. This behavior is intuitive. A distance function involves finding the nearest point, while a smoothed distance instead considers nearby regions. Also, these samples are assigned lower weights, which compensates for their higher sampling density.

where the clamp prevents the logarithm from receiving values that are too small, and the absolute value avoids negative predictions, which commonly occur at early stages of training due to potential negative reweighting at initialization.All design choices, including the offset term, loss form, and function $\mathcal{G}$, are validated in Fig. 44.

Results. Figure 45 compares our approach with directly applying various quadrature rules to the inner integral and then applying the logarithm. All variants of our method significantly outperform these baselines, and the offset term further reduces the MSE. Figure 46 reports the squared bias of all methods. Although Monte Carlo estimators are unbiased for the inner integral, directly applying a nonlinear transformation leads to large bias. In contrast, our methods are inherently biased, but achieve much lower bias due to more accurate predictions with lower variance.

E.2 Towards nonlinear PDE

Since our integrator is able to learn the effect of the Poisson kernel, we conjecture that it can also handle nonlinear PDEs within a similar WoS framework. For the p-Laplacian, the solution admits a nonlinear

Table 11. Generalized winding numbers, Results on Test Sets. We report the mean squared error (MSE) of all methods using 32 samples per pixel on the test set. The test geometries and boundary conditions are generated in the same manner as the training set, but are not used during training. The best result is shown in **bold** and the second best is underlined. One of the test scene is also visualized in the figure above. Our integrator-only network already significantly outperforms all baseline methods, and the jointly trained estimator can even achieve an 100–300× reduction in MSE.

				Ours			
	MC rand	MC strat	Trapz strat	INet rand	INet strat	INet SNet	×
0	3.61e-1	9.99e-1	1.00e+0	2.65e-2	5.23e-3	**2.17e-3**	166.4×
1	3.08e-1	3.49e-1	3.57e-1	2.65e-2	6.48e-3	**2.57e-3**	120.0×
2	3.65e-1	2.27e-1	2.27e-1	1.49e-2	4.21e-3	**1.71e-3**	214.0×
3	2.86e-1	2.31e-1	2.34e-1	2.45e-2	6.73e-3	**3.48e-3**	82.3×
4	3.59e-1	1.02e-1	1.01e-1	1.78e-2	5.23e-3	**1.18e-3**	305.3×
5	3.43e-1	2.03e-1	2.04e-1	2.27e-2	5.21e-3	**2.28e-3**	150.5×

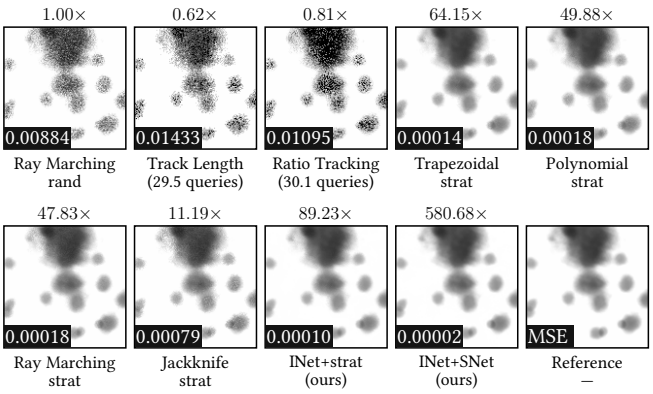


Fig. 48. Transmittance, Results on Test Sets. We compare the MSE of transmittance estimates using different integrating and sampling strategies. Ray marching–based methods use 24 density queries per pixel, while track-length and ratio tracking use an average of 29 and 30 queries per pixel, respectively. Trapezoidal and polynomial rules are applied on top of ray marching, and all stratified sampling methods use uniformly jittered samples. Our estimator achieves significantly lower MSE than all baselines. A complete table including additional test scenes is provided in Table 4.

boundary potential representation

$$u(x) = \int_{\partial\Omega} \Phi_p(x,y)\,|\nabla u(y)|^{p-2}\,\partial_n u(y)\,d\sigma(y), \tag{57}$$

where $\Phi_p(x,y) \sim |x-y|^{\frac{p-n}{p-1}}$ is the fundamental solution of the p-Laplacian. In the special case of $p = 2$,

$$u(x) = \int_{\partial\Omega} \Phi_2(x,y)\,\partial_n u(y)\,d\sigma(y), \tag{58}$$

which reduces to the standard Laplacian with a fixed Poisson kernel. For $p \neq 2$, however, the effective kernel in Eq. (57) depends on the unknown solution itself through the factor $|\nabla u(y)|^{p-2}$. This yields

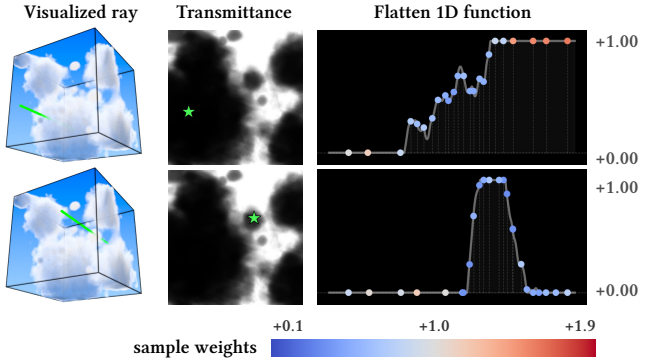


Fig. 49. Transmittance, Learned strategies. We visualize two marching rays through the volume: one passing through a thick dense cloud and the other intersecting only a small region. Samples are sparse in empty regions and are automatically concentrated in dense or rapidly varying regions.

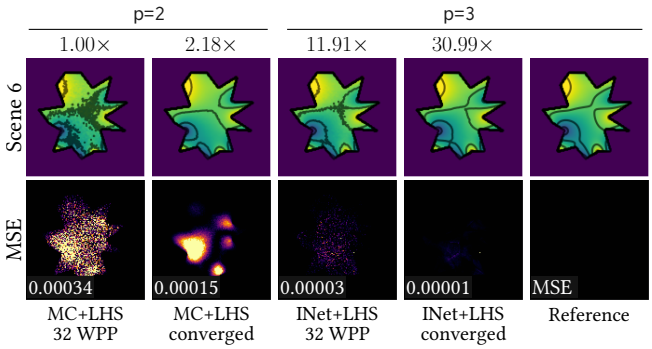


Fig. 50. WoS, Solving p-Laplacian. We demonstrate that our neural integrator can handle nonlinear differential operators by learning to solve the p-Laplacian with $p = 3$. In contrast, standard Monte Carlo WoS methods are typically limited to the linear Laplacian case ($p = 2$). Using the $p = 2$ solution directly for the case $p \neq 2$ can lead to large bias, even after convergence. With the same set of 32 walks per pixel, our learned integrator produces results with low noise and accurately captures the 2nd order effects.

a nonlinear, solution-dependent integral operator, which cannot be handled by classical Monte Carlo WoS. However, as shown in Fig. 50, when trained with reference solutions from a FEM solver, our estimator can effectively handle this nonlinear problem using the same geometric queries as in the WoS framework and remains grid-free. The integrator learns to infer the effective kernel from all observations, which highlights another potential of our framework: even when the integrand is not explicitly known and only integral solutions are available, the neural integrator can learn to infer the underlying integral operator directly from data through training.

F TIMING

In Table 15, Table 16, and Table 17, we report the average runtime of different methods across resolutions and sample counts. Monte Carlo traces all paths in a single pass and is therefore the cheapest. The INet-only variant also uses a single tracing pass, but must additionally record auxiliary information, incurring higher memory

Table 12. Transmittance, MSE comparison. We compare our method against biased ray marching [Kettunen et al. 2021] (which typically attains lower MSE than its unbiased variant), track-length [Coleman 1968; Novák et al. 2018], ratio tracking [Novák et al. 2014], the trapezoidal rule and regression-based Monte Carlo [Salaün et al. 2022] using degree-3 polynomials on ray marching, as well as the Jackknife estimator [Peters 2025]. For ray-marching-based approaches, including ours, we use one ray with 24 queries per pixel. For tracking-based approaches, we continue sampling until more than 24 queries have been performed in previous tracking. For all algorithms, we adopt the simplest setup with a global majorant of 1 and no importance-sampling structures [Kettunen et al. 2021; Peters 2025], which are difficult to construct for our procedural noise fields. Our approach shows significant and robust improvements over all baselines.

								Ours		
Scene	Ray Marching rand	Track-length	Ratio tracking	Trapz strat	Poly strat	Ray Marching strat	Jackknife strat	INet strat	INet SNet	×
Test 0	6.01e-3	1.52e-2	7.61e-3	1.28e-4	1.54e-4	1.19e-4	5.35e-4	9.55e-5	**2.14e-5**	281.4×
Test 1	7.73e-3	1.81e-2	9.37e-3	1.25e-4	1.57e-4	1.59e-4	6.71e-4	1.07e-4	**2.87e-5**	269.4×
Test 2	4.25e-3	1.44e-2	5.90e-3	1.02e-4	1.20e-4	9.34e-5	4.22e-4	8.41e-5	**3.10e-5**	137.0×
Test 3	8.84e-3	1.43e-2	1.09e-2	1.38e-4	1.77e-4	1.85e-4	7.90e-4	9.91e-5	**1.52e-5**	580.7×
Test 4	7.72e-3	1.90e-2	1.00e-2	1.59e-4	1.89e-4	2.26e-4	8.32e-4	1.19e-4	**2.22e-5**	347.6×
Test 5	1.04e-2	2.14e-2	1.25e-2	1.48e-4	1.81e-4	1.56e-4	7.38e-4	1.26e-4	**2.50e-5**	414.3×
Donut	6.67e-3	1.63e-2	8.46e-3	8.21e-5	8.81e-5	8.57e-5	3.79e-4	7.75e-5	**1.30e-5**	515.2×
Letters	5.33e-3	1.10e-2	7.27e-3	5.84e-5	6.49e-5	5.82e-5	2.60e-4	1.85e-4	**3.99e-5**	133.6×

Table 13. Image quality comparison on different scenes. We report LPIPS (↓), PSNR (↑), and DSSIM (↓) for Monte Carlo (MC) and our method.

	LPIPS ↓		**PSNR ↑**		**DSSIM ↓**	
Scene	MC	Ours	MC	Ours	MC	Ours
scene0	9.6e-05	**4.8e-05**	22.689945	**28.622837**	0.001844	**0.000364**
scene4	5.36e-04	**4.7e-05**	12.927814	**24.37705**	0.020865	**0.001468**
whiteroom	7e-06	**4e-06**	27.25726	**31.471718**	0.003855	**0.001345**
livingroom	9e-06	**5e-06**	24.588661	**30.209072**	0.065225	**0.027199**
staircase	4e-06	**3e-06**	28.659288	**31.098501**	0.005401	**0.002469**

Table 14. WoS, Results on Test Sets. We report the MSE of all methods using 32 walks per pixel on the test set. The test geometries and boundary conditions are generated in the same way as the training set, but are not used during training. The best result is shown in **bold** and the second best is underlined. The highlighted test scene is also visualized in the figure above. Our approach achieves a significant improvement over all baselines.

				Ours			
	MC rand	MC LHS	Trapz LHS	INet rand	INet LHS	INet SNet	×
0	3.04e-4	2.10e-4	2.95e-4	2.40e-5	1.94e-5	**8.93e-6**	34.09×
1	2.16e-4	1.47e-4	2.02e-4	2.18e-5	1.82e-5	**9.74e-6**	22.24×
2	8.84e-5	5.94e-5	8.63e-5	1.45e-5	1.13e-5	**4.75e-6**	18.61×
3	4.34e-4	2.86e-4	3.84e-4	3.22e-5	2.83e-5	**1.61e-5**	26.92×
4	2.33e-4	1.48e-4	2.11e-4	1.91e-5	1.66e-5	**6.35e-6**	36.72×
5	2.62e-4	1.71e-4	2.42e-4	2.27e-5	1.79e-5	**9.91e-6**	26.46×

write bandwidth, and then apply a bidirectional Transformer, which introduces substantial computational overhead. The INet+SNet variant is the most expensive: for N samples, it alternates iteratively between the path tracer and the network, generating one sample and tracing one ray at a time. This process involves more network invocations as well as additional memory synchronization and kernel launches, leading to significantly higher runtime. It may be possible to integrate both tracing and attention into a fused tiny neural network call [Müller et al. 2021], thereby reducing synchronization.

In Table 18, we report the overhead of the generalized winding number application, where all sample queries are implemented in custom GPU kernels and are therefore very fast. The runtimes of other 1D applications should be comparable, as they use neural networks of similar size. On the other hand, for the direct illumination example we employ a relatively large network, which is likely unnecessarily large for many simple 1D tasks. In practice, this leaves substantial room for further optimization and network distillation.

In Table 19, we report the runtime of the Walk-on-Spheres application. Although the INet has a similar size, SNet is even more expensive in this setting, as it must interact not only with the sample query once per sample (walk) but also at every step within each walk, and additionally involves an RNN invocation, as illustrated in Fig. 30. Again, we have not tuned the network size or applied sophisticated performance optimizations, and therefore believe that there remains substantial room for further improvement.

Table 15. Direct Illumination, Timing comparison at different resolutions, 16 SPP. Average render time per frame (ms) over 50 runs on RTX 5090.

Scenes	Resolution								
	128^2			64^2			32^2		
	MC	INet	INet+SNet	MC	INet	INet+SNet	MC	INet	INet+SNet
TestScene 1	9.1212	71.5792	126.0608	3.9098	16.8365	52.3890	2.4705	4.5066	37.4544
TestScene 2	8.7003	71.4898	108.0182	3.7251	16.4477	47.3715	2.3906	4.3850	35.9349
TestScene 3	8.9116	71.5304	163.5606	3.7257	16.3384	59.9288	2.4461	4.3650	38.5036
TestScene 4	8.6036	71.6012	147.2001	3.8980	16.4516	55.1738	2.5353	4.4765	37.7700
TestScene 5	8.5622	71.5769	171.0040	3.7393	16.3335	61.6839	2.4418	4.2839	39.5833
LivingRoom	8.9416	71.7508	205.7233	3.7442	16.5000	66.6557	2.6066	4.5445	41.2026
WhiteRoom	9.7791	72.3309	203.5018	3.9949	16.7013	68.1408	2.6507	4.4997	40.9445
Staircase	9.1982	71.7761	202.8882	3.8178	16.6014	68.9702	2.5540	4.5384	40.0439
VeachMIS	8.8394	71.6329	185.4065	3.7912	16.4635	63.8202	2.4409	4.3882	39.2754
Teaser	9.5032	71.9758	202.7156	3.9020	16.5396	68.3487	2.5590	4.7767	40.9541

Table 16. Direct Illumination, Timing comparison at different resolutions, 8 SPP. Average render time per frame (ms) over 50 runs on RTX 5090.

Scenes	Resolution								
	128^2			64^2			32^2		
	MC	INet	INet+SNet	MC	INet	INet+SNet	MC	INet	INet+SNet
TestScene 1	5.4885	41.1454	63.1500	2.8220	8.9501	25.7662	2.0231	3.2834	18.9646
TestScene 2	5.4233	41.0846	52.7541	2.7082	8.8804	24.1346	2.0828	3.1812	18.4353
TestScene 3	5.5802	41.1196	81.2445	2.7919	8.9891	29.4524	1.9518	3.1986	19.9745
TestScene 4	5.4861	41.1487	70.5639	2.8916	9.4566	27.6677	2.0611	3.5245	19.4704
TestScene 5	5.5372	40.9904	85.2704	2.7815	8.8196	29.6973	2.1159	3.3008	19.4707
LivingRoom	5.5415	41.2244	100.9542	2.8718	9.0220	32.6441	2.1376	3.2723	20.4393
WhiteRoom	5.8787	41.3373	101.4160	2.9283	9.0514	33.3967	2.1000	3.3081	19.9457
Staircase	5.7230	41.1986	100.9698	2.8860	8.9986	32.8409	2.2111	3.3166	19.7175
VeachMIS	5.6926	41.2114	92.0142	2.6993	9.2307	31.3066	2.1293	3.3619	19.6291
Teaser	5.6371	41.2829	100.3672	2.9428	8.9032	33.1569	2.1394	3.2960	20.4764

Table 17. Direct Illumination, Timing comparison at different resolutions, 4 SPP. Average render time per frame (ms) over 50 runs on RTX 5090.

Scenes	Resolution								
	128^2			64^2			32^2		
	MC	INet	INet+SNet	MC	INet	INet+SNet	MC	INet	INet+SNet
TestScene 1	3.8563	23.6484	31.0031	2.2941	6.1851	13.0255	1.8652	2.9466	9.9359
TestScene 2	3.7746	23.8817	26.2957	2.3115	6.1548	12.6053	1.8963	2.8624	9.5361
TestScene 3	3.8333	23.6003	39.9498	2.8260	6.3972	15.2304	1.9167	2.7086	10.1630
TestScene 4	3.6378	23.6981	34.3874	2.4022	6.1740	13.9926	1.9160	2.6796	9.8214
TestScene 5	3.6770	23.6267	41.5254	2.2670	6.2218	15.5387	1.8795	2.6880	9.8639
LivingRoom	3.8049	23.8030	49.3197	2.4002	6.3749	16.6794	1.8553	2.7423	10.1617
WhiteRoom	3.8466	23.9456	49.3738	2.3084	6.6511	16.6683	1.9620	2.8212	10.2996
Staircase	3.8341	23.7278	49.2780	2.4438	6.3797	16.7652	1.9086	2.7541	10.2994
VeachMIS	3.7817	23.6381	44.9722	2.3469	6.1743	15.8584	1.9000	2.7230	9.7852
Teaser	3.9815	23.7801	49.1862	2.3673	6.3590	16.8467	1.9421	2.7024	10.4317

Table 18. Generalized Winding Number, Average render time (ms) for different SPP and resolutions, on RTX 5090.

SPP	Method								
	MC			INet			INet+SNet		
	32^2	64^2	128^2	32^2	64^2	128^2	32^2	64^2	128^2
32	0.0374	0.0375	0.0435	5.4363	10.7975	46.6305	49.7708	65.0407	237.5646
16	0.0274	0.0285	0.0762	5.3793	6.7112	30.2862	27.6903	32.7452	120.0095
8	0.0265	0.0285	0.0269	5.4165	5.7801	19.3046	15.7227	17.2114	61.2564
4	0.0260	0.0275	0.0319	5.3593	5.7216	14.6914	10.4049	11.7134	32.5794

Table 19. WoS, Average render time (ms) for different SPP and resolutions, on RTX 4090.

SPP	Method								
	MC			INet			INet+SNet		
	32^2	64^2	128^2	32^2	64^2	128^2	32^2	64^2	128^2
32	2.0581	10.2219	43.9332	7.0513	39.1373	160.2633	176.8868	288.9736	855.6341
16	0.9054	5.0262	21.8564	8.9474	18.4967	89.4162	90.2725	141.8054	429.5298
8	0.4682	1.3719	10.6536	6.2745	10.4266	52.6012	46.5906	69.9915	216.4006
4	0.3079	0.7323	4.9029	8.3666	8.0049	32.5625	24.8410	38.2830	109.4693