Neural Control Variates

THOMAS MÜLLER, NVIDIA
FABRICE ROUSSELLE, NVIDIA
ALEXANDER KELLER, NVIDIA
JAN NOVÁK, NVIDIA
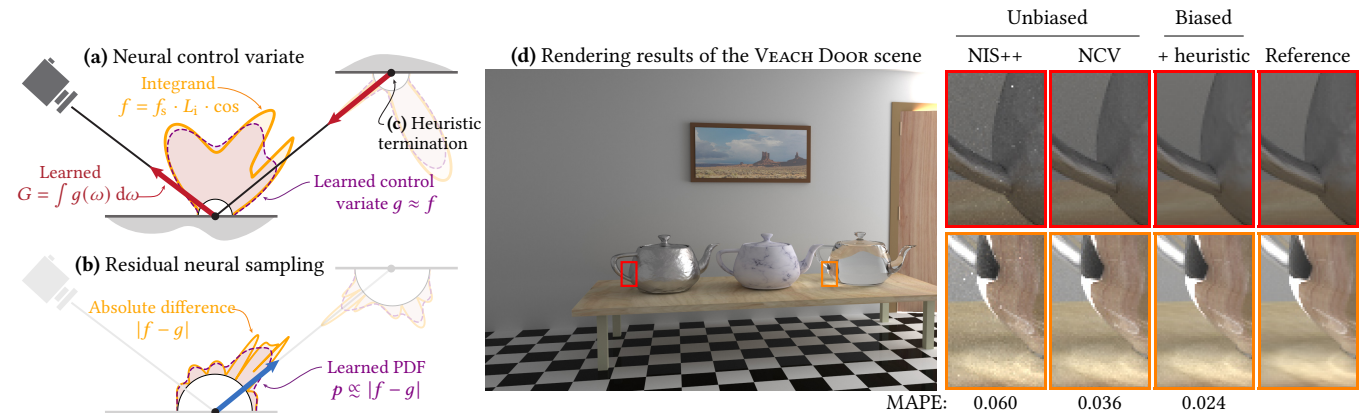
Fig. 1. When applied to light-transport simulation, our neural control-variate algorithm **(a)** learns an approximation G of the scattered light field and corrects the approximation error by estimating the difference between the original integrand f and the corresponding learned control variate g. This is enabled by our construction that couples g and G such that g always exactly integrates to G. To further reduce noise, we importance sample the absolute difference $|f - g|$ using a learned probability density function (PDF) p **(b)**. We also provide a heuristic **(c)** to terminate paths without estimating the difference. This reduces the mean path length and removes most of the remaining noise. On the right **(d)**, we compare the error of rendering the VEACH DOOR scene using an improved variant (NIS++) of neural importance sampling [Müller et al. 2019] to our neural control variates (NCV) with and without our path termination heuristic.

We propose neural control variates (NCV) for unbiased variance reduction in parametric Monte Carlo integration. So far, the core challenge of applying the method of control variates has been finding a good approximation of the integrand that is cheap to integrate. We show that a set of neural networks can face that challenge: a normalizing flow that approximates the shape of the integrand and another neural network that infers the solution of the integral equation. We also propose to leverage a neural importance sampler to estimate the difference between the original integrand and the learned control variate. To optimize the resulting parametric estimator, we derive a theoretically optimal, variance-minimizing loss function, and propose an alternative, composite loss for stable online training in practice. When applied to light transport simulation, neural control variates are capable of matching the state-of-the-art performance of other unbiased approaches, while providing means to develop more performant, practical solutions. Specifically, we show that the learned light-field approximation is of sufficient quality for high-order bounces, allowing us to omit the error correction and thereby dramatically reduce the noise at the cost of negligible visible bias.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Ray tracing**; *Supervised learning by regression*; • **Mathematics of computing** → *Sequential Monte Carlo methods*.

1 INTRODUCTION

Monte Carlo (MC) integration is a simple numerical recipe for solving complicated integration problems. The main drawback of the straightforward approach is the relatively slow convergence rate that manifests as high variance of MC estimators. Hence, many approaches have been developed to improve the efficiency. Among the most frequently used ones are techniques focusing on carefully placing samples, e.g. antithetic sampling, stratification, quasi-random sampling, or importance sampling. A complimentary way to further reduce variance is to leverage hierarchical integration or the concept of control variates. In this article, we focus on the latter approach and present parametric control variates based on neural networks.

Reducing variance by control variates (CV) amounts to leveraging an approximate solution of the integral corrected by an estimate of the approximation error. The principle is given by the following identity:

$$F = \int_{\mathcal{D}} f(x) \, dx = \alpha \cdot G + \int_{\mathcal{D}} f(x) - \alpha \cdot g(x) \, dx. \qquad (1)$$

Instead of integrating the original function f to obtain the solution F, we leverage an α-scaled approximation G, that corresponds to integrating a (different) function g—the *control variate*—over the

same domain $\mathcal{D}$, i.e. $G = \int_{\mathcal{D}} g(x) \, \mathrm{d}x$ (we drop $\mathcal{D}$ in the rest of this section for brevity). The approximation error is corrected by adding an integral of the difference $f(x) - \alpha \cdot g(x)$; this makes the right-hand side equal to the left-hand one.

The numerical efficiency of estimating the right-hand side, relative to estimating the original integral, depends on the scaled control variate making the integration easier, for example by making the integrand smoother as illustrated in Figure 2. This will typically be the case as long as f and g are (anti-)correlated. In fact, the scaling coefficient α, which controls the strength of applying the CV, should be derived from the correlation of the two functions. In a nutshell, a successful application of control variates necessitates a g that approximates the integrand f sufficiently well, and permits an efficient evaluation and computation of G and α.

In this work, we propose to infer the control variate g from observations of f using machine learning. Since the control variate is learned, the main challenge becomes representing it in a form that permits (efficiently) computing its integral, $G = \int g(x) \, \mathrm{d}x$. We propose to sidestep this integration problem by introducing a CV model that satisfies $G = \int g(x) \, \mathrm{d}x$ *by construction*: we decompose the control variate $g(x)$ into its normalized form—the shape $\bar{g}(x)$— and its integral G, such that $g(x) = \bar{g}(x) \cdot G$. The shape and the integral can be modeled independently. We infer the integral G and the coefficient α using one neural network for each. For the shape $\bar{g}$, we leverage a tailored variant of normalizing flows, which are capable of representing normalized functions. The parameters of the flow are inferred using a set of neural networks.

When the control variate is designed well, the residual integral $\int f(x) - \alpha \cdot g(x) \, \mathrm{d}x$ carries less energy than the original integral $\int f(x) \, \mathrm{d}x$. However, the residual *integrand* may feature shapes that are hard to sample with hand-crafted distributions; this is why many prior works in graphics did not demonstrate significant efficiency gains when combining control variates with importance sampling.

We address this by employing neural importance sampling (NIS) as proposed by Müller et al. [2019] that is capable of importance sampling arbitrary integrands, including the residual ones in our case. We show that an estimator that utilizes both techniques, NCV and NIS, features the strengths of each approach as long as all trainable parameters are optimized jointly; to that end we derive two loss functions, one theoretically optimal and one that yields robust optimization in practice.

We demonstrate the benefits of neural control variates on light-transport simulations governed by Fredholm integral equations of the second kind. These are notoriously difficult to solve efficiently due to their recursive nature, often necessitating high-dimensional samples in the form of multi-vertex transport paths (obtained using e.g. path tracing). In this context, control variates offer two compelling advantages over prior works that only focus on placing the samples. First, control variates reduce the number of constructed path vertices as the difference integral typically carries less energy than the original integral. Paths can thus be terminated earlier using the learned scattered radiance as an approximation of the true scattered radiance; we propose a heuristic that minimizes the resulting bias. Second, control variates trivially support spectrally resolved path tracing by using a different g for each spectral band. To avoid
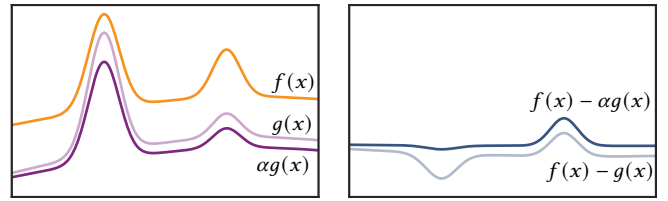

Fig. 2. Computing an integral F of a function $f(x)$ with the help of a control variate $g(x)$ (left) amounts to using a known (or efficient-to-compute) integral $G = \int g(x) \, \mathrm{d}x$ and adding the integrated difference $f(x) - g(x)$ (right). The overall variance may be reduced by minimizing the variance of the difference using an α-scaled control variate, where $\alpha = \mathrm{Cov}(f, g) / \mathrm{Var}(g)$.

computational overhead of using potentially *many* control variates, we develop a novel normalizing flow that can represent multiple (ideally correlated) control variates at once. Spectral noise, which is typical for importance sampling that only targets scalar distributions, is thus largely suppressed. The benefits are clearly notable in several of our test scenes.

In summary, we present the following contributions:

- a tractable neural control variate modeled as the product of normalized shape $\bar{g}$ and the integral G,
- a multi-channel normalizing flow for efficient handling of spectral integrands,
- an estimator combining neural control variates with neural importance sampling, for which we present
- a variance-optimal loss derived from first principles, and an empirical composite loss that yields stable online optimization on noisy estimates of $f(x)$, and finally,
- a practical light-transport simulator that heuristically omits estimating the residual integral when visible bias is negligible.

2 RELATED WORK

The application of control variates has been explored in many fields, predominantly in the field of financial mathematics and operations research, see [Broadie and Glasserman 1998; Hesterberg and Nelson 1998; Kemna and Vorst 1990] for examples. Later on, Glynn and Szechtman [2002] focused on relating the concept to antithetic sampling, rotation sampling, and stratification, among other techniques. In computer graphics, Rousselle et al. [2016] link control variates to solving the Poisson equation in screen space and Kondapaneni et al. [2019] use the concept to interpret their optimally weighted multiple-importance sampler.

Since a poorly chosen control variate may even decrease efficiency, early research focused on an efficient and accurate estimation of the scaling coefficient α. While the optimal, variance-minimizing value of α is known to be $\mathrm{Cov}(f, g) / \mathrm{Var}(g)$ (see Figure 2 for an illustration), estimating it numerically may introduce bias if done using samples correlated to the samples used for the actual estimate [Lavenberg et al. 1982; Nelson 1990]. We resolve this issue by providing recipes for obtaining α that do not bias the estimator.

We are not the first to apply control variates to light transport simulation. Lafortune and Willems successfully leveraged CVs based

on ambient illumination [1994] and hierarchically stored radiance values [1995] to accelerate the convergence of path tracing. Pegoraro et al. [2008a,b] applied a similar idea to volumetric path tracing, but were restricted to near-isotropic volumes. Fan et al. [2006] and Kondapaneni et al. [2019] use a linear combination of multiple importance-sampling densities as a control variate, which is an adaptation of a technique by Owen and Zhou [2000]. Others proposed to apply CVs to carefully chosen subproblems, such as estimating direct illumination [Clarberg and Akenine-Möller 2008; Szécsi et al. 2004], sampling free-flight distances in participating media [Georgiev et al. 2019; Novák et al. 2014; Szirmay-Kalos et al. 2011], or unbiased denoising and re-rendering [Rousselle et al. 2016; Yamaguchi et al. 2018].

One of the challenges of successfully applying control variates is an efficient estimation of the residual integral $\int f(x) - g(x) \, dx$; this is typically harder than (importance) sampling f alone. We demonstrate that parametric trainable control variates can be well complemented by trainable importance samplers (such as neural importance sampling [Müller et al. 2019]) yielding better results than each technique in isolation.

Multi-level Monte Carlo integration. Heinrich [1998; 2000] proposed to apply the CV concept in a hierarchical fashion: Each successive estimator of a difference improves the estimate of its predecessor. This technique is known as multi-level Monte Carlo integration and it has been applied in stochastic modeling [Giles 2008], solving partial differential equations [Barth et al. 2011], or image synthesis [Keller 2001]; see the review by Giles [2013] for other applications. While the allocation of samples across the estimators is key to efficiency, classic representations of functions quickly render the approach intractable in higher dimensions.

Realistic image synthesis with neural networks. Similar to Monte Carlo methods for high-dimensional integration, neural networks are especially helpful in high-dimensional approximation. In computer graphics, they have been shown very suitable for compressing and inferring fields of radiative quantities (or their approximations) in screen space [Nalbach et al. 2017], on surfaces [Maximov et al. 2019; Ren et al. 2013; Thies et al. 2019; Vicini et al. 2019], on point clouds [Hermosilla et al. 2019], or in free space [Kallweit et al. 2017; Lombardi et al. 2019; Meka et al. 2019; Sitzmann et al. 2018]; see the survey by [Tewari et al. 2020] for additional examples. These approaches are largely orthogonal to our technique. In fact, many of these ideas may improve the learning and representation of the approximate solution G in specific situations. For instance, one could employ voxel grids with warping fields instead of multi-layer perceptrons [Lombardi et al. 2019], combat overfitting using miplevel hierarchies [Thies et al. 2019], or handle scene partitions using dedicated networks [Ren et al. 2013]; shall the application need it. Leaving these as possible future extensions, we instead focus on a shortcoming that is common to all the aforementioned approaches: occasional deviations from the ground-truth solution observable as e.g. patchiness, loss of contrast, or dull highlights. We propose to correct the errors using the mechanism of control variates, i.e. we add an estimate of the difference between the correct solution and

the approximation to recover unbiased results with error manifesting merely as noise. We view our neural control variates as a step towards bringing data-driven and physically-based rendering closer.

Normalizing flows. Normalizing flows [Tabak and Turner 2013; Tabak and Vanden Eijnden 2010] are a technique for mapping arbitrary distributions to a base distribution; e.g. the normal distribution. The mappings are formally obtained by chaining an infinite series of infinitesimal transformations, hence the name flow. The technique has been successfully leveraged for variational inference, either in the continuous form [Chen et al. 2018] or as a finite sequence of warps [Dinh et al. 2014; Rezende and Mohamed 2015]. Numerous improvements followed soon after: the modeling power of individual transforms has been enhanced using non-volume preserving warps [Dinh et al. 2016], piecewise-polynomial warps [Müller et al. 2019], or by injecting learnable 1×1 convolutions between the warps [Kingma and Dhariwal 2018]. Others have demonstrated benefits by formulating the estimation autoregressively [Huang et al. 2018; Kingma et al. 2016; Papamakarios et al. 2017]; we refer the reader to the surveys by Kobyzev et al. [2019]; Papamakarios et al. [2019] for an introduction and comparisons of different approaches.

In light transport simulation, Zheng and Zwicker [2019] and Müller et al. [2019] leverage modified normalizing flows to learn and sample from parametric distributions. In analogy, we use our multi-channel flow to represent the spectrally resolved per-channel normalized form $\bar{g}$ of the control variate g.

(Neural) Control Variates based on Stein's identity and beyond. Assaraf and Caffarel [1999] suggest representing the control variate in terms of the score function $s(x) = \nabla \log p(x)$, where $p(x)$ is the importance-sampling density. The score function has zero expectation, i.e. $\mathbb{E}_p[s(x)] = 0$, trivially allowing its use as a control variate of a stochastic estimator. Through Stein's [1972] identity, the score function can be reparameterized to act as an effective control variate. Many such reparameterizations were proposed, be they parametric polynomials [Assaraf and Caffarel 1999; Mira et al. 2013], non-parametric [Oates et al. 2014], or parameterized by neural networks [Grathwohl et al. 2018; Wan et al. 2019].

In contrast to our use of normalizing flows, using the score function and Stein's identity as a control variate has one major limitation: the integral of the control variate G is unknown—one only knows that the *expectation* of the control variate under samples from p is zero. This limitation results in the following practical shortcomings: (i) it is not possible to use the CV integral G as a light-field approximation in the way we propose, and (ii) it is difficult to adapt the sampling density p to the control variate; optimizing the sampling density to importance sample the residual difference $|f - g|$ would alter the score function and thereby the control variate, creating a circular dependency. In future work, it may be possible to derive a joint optimization between score-function-based control variates and importance sampling similar to our unbiased variance loss. Beyond Stein's identity, neural networks were also used control variates based on the Martingale representation theorem for solving partial differential equations [Vidales et al. 2018].

3 PARAMETRIC TRAINABLE CONTROL VARIATES

In this section, we propose a novel model for trainable control variates in the context of integro-approximation: our goal is to reduce the variance of estimating the parametric integral

$$F(y) = \int_{\mathcal{D}} f(x, y) \, dx$$
$$= \alpha(y) \cdot G(y) + \int_{\mathcal{D}} f(x, y) - \alpha(y) \cdot g(x, y) \, dx \quad (2)$$

parameterized by y using the control variate g. This means that we need to represent and approximate functions besides computing integrals. For instance, in the light transport application of Section 6, $F(y)$ is the reflected radiance and the parameter y represents the reflection location and direction.

In many applications and especially in computer graphics, the functions $F(y)$ and $f(x, y)$ may have infinite variation and lack smoothness. Their models thus need to be sufficiently flexible and highly expressive. Therefore, we make the design decision to model the CV using neural networks driven by an optimizable set of parameters θ_g; a discussion of alternatives is deferred to Section 8.

Tractable neural control variates. In order to use Equation (2), the neural model must permit an efficient evaluation of the control variate $g(x, y; \theta_g)$ and its integral $G(y) = \int g(x, y; \theta_g) \, dx$. This turns out to be the key challenge. Modeling g using a neural network may be sufficiently expressive, but computing the integral G would require some form of numerical integration necessitating multiple forward passes to evaluate $g(x, y; \theta_g)$; a cost that is too high.

We avoid this issue by restricting ourselves to functions where the integral is known. Specifically, we consider *normalized* functions that integrate to 1. Arbitrary integrands can still be matched by scaling the normalized function by a (learned) factor. Hence, our parametric control variate

$$g(x, y; \theta_g) := \bar{g}(x, y; \theta_{\bar{g}}) \cdot G(y; \theta_G) \quad (3)$$

is defined as the product of two components: a parametric normalized function $\bar{g}(x, y; \theta_{\bar{g}})$ and a parametric scalar value $G(y; \theta_G)$. From now on, we refer to $\bar{g}$ and G as the *shape* and the *integral* of the CV, each of which is parameterized by its own set of parameters and $\theta_g := \theta_{\bar{g}} \cup \theta_G$. This decomposition has the advantage that computing the integral G amounts to evaluating a neural network once, rather than performing a costly numerical integration of $g(x, y; \theta_g)$ that requires a large number of network evaluations.

The rest of this section proposes parametric models for the shape (Section 3.1), the integral (Section 3.2), and the coefficient (Section 3.3) of the control variate. Sections 4 and 5 then describe an efficient combination with a parametric importance sampler and the optimization of all trainable parameters.

3.1 Modeling the Shape of the Control Variate

We now address the main challenge of modeling CVs using neural networks: learning normalized functions that we use to represent the shape $g(x, y; \theta_g)$ of the CV. Normalizing the output of a neural network is generally difficult. We thus resort to a class of models where the network output is used to merely parameterize a transformation, which can be used to warp a function without changing its

integral. This allows for learning functions that are normalized *by construction*. Such models are referred to as normalizing flows (see e.g. [Kobyzev et al. 2019; Papamakarios et al. 2019]). In what follows, we briefly review the concept of normalizing flows and discuss the details of using them to learn the shape of the CV.

Normalizing flow preliminaries. A normalizing flow is a differentiable, multi-dimensional, compound mapping for transforming probability densities. The mapping $\hat{h} = h_L \circ \cdots \circ h_2 \circ h_1$ comprises L bijective warping functions; it is therefore also bijective as a whole. The warping functions $h: \mathcal{X} \to \mathcal{X}'$ induce a density change according to the change-of-variables formula

$$p_{\mathcal{X}'}(x') = p_{\mathcal{X}}(x) \cdot \left| \det\left(\frac{\partial h(x)}{\partial x^T} \right) \right|^{-1}, \quad (4)$$

where p is a probability density, $x \in \mathcal{X}$ is the argument of the warp, $x' = h(x) \in \mathcal{X}'$ is the output of the warp, and $\left(\frac{\partial h(x)}{\partial x^T} \right)$ is the Jacobian matrix of h at x.

The density change induced by a chain of L such warps can be obtained by invoking the chain rule. This yields the following product of absolute values of Jacobian determinants:

$$J(x) = \prod_{i=1}^{L} \left| \det\left(\frac{\partial h_i(x_i)}{\partial x_i^T} \right) \right|, \quad (5)$$

where now $x_1 \equiv x$. The i-th term in the product represents the absolute value of the Jacobian determinant of the i-th warp with respect to the output of warp $i - 1$.

The transformed variable $\hat{x} = \hat{h}(x)$ is often referred to as the latent variable in latent space $\mathcal{L}$. Its distribution is related to the distribution of the input variable by combining Equations (4,5):

$$p_{\mathcal{L}}(\hat{x}) = \frac{p_{\mathcal{X}}(x)}{J(x)}. \quad (6)$$

The distribution of latent variables $p_{\mathcal{L}}(\hat{x})$ is typically chosen to be simple and easy to sample; we use the uniform distribution $p_{\mathcal{L}}(\hat{x}) \equiv p_{\mathcal{U}}(\hat{x})$ over the unit hypercube.

In order to achieve high modeling power, neural normalizing flows utilize *parametric* warps that are driven by the output of neural networks. To allow for modeling correlations across dimensions, the outputs of individual warps need to be fed into neural networks conditioning the subsequent warps in the flow. In the context of probabilistic modeling, two main approaches have been proposed to that end: autoregressive flows [Huang et al. 2018; Kingma et al. 2016; Papamakarios et al. 2017; Rezende and Mohamed 2015] and coupling flows [Dinh et al. 2014, 2016; Müller et al. 2019]. Both of these approaches yield flows that are (i) invertible, (ii) avoid the cubic cost of computing determinants of dense Jacobian matrices, and (iii) avoid the need to differentiate through the neural network to compute relevant entries in the Jacobian.

In this work, efficient invertibility of the flow is not needed as modeling the CV shape requires evaluating the flow in only one direction. However, we still take advantage of the previously proposed autoregressive formulation to ensure tractable Jacobian determinants. Furthermore, we show that the model can be further accelerated in cases when multiple densities—specifically, multiple channels of the control variate—are being learned.
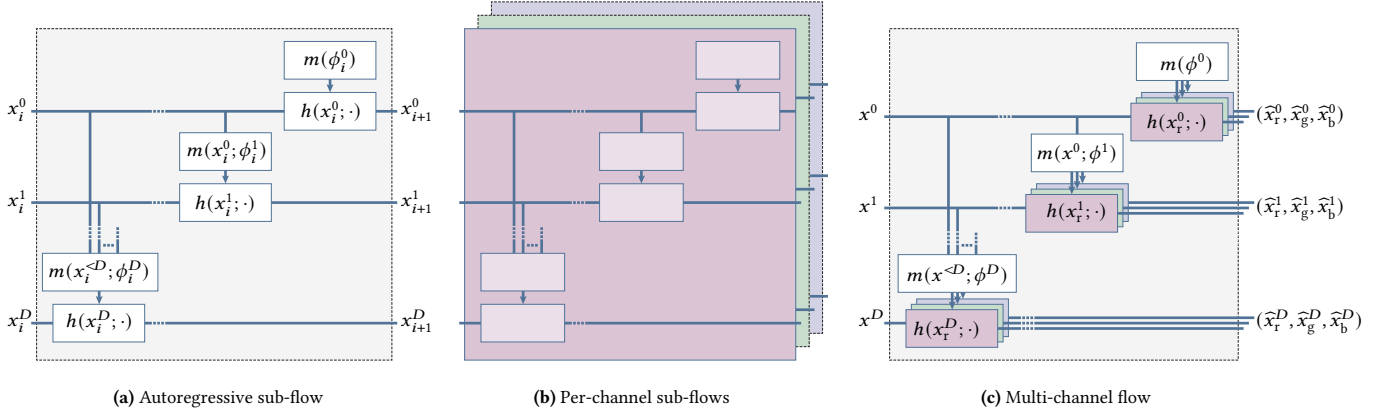
(a) Autoregressive sub-flow   **(b)** Per-channel sub-flows   **(c)** Multi-channel flow

Fig. 3. We model the shape of the control variate using bijective transformations assuming autoregressive structure **(a)** as proposed by Kingma et al. [2016] in the context of probabilistic generative models. Concatenating multiple autoregressive blocks (sub-flows) increases the expressivity of the model. To handle multiple control variates (e.g. one for each color channel), one can instantiate a flow for each "channel" **(b)**; we avoid repeating the expressions in (b) for brevity; the only difference to the left illustration is that all x and ϕ would have a *channel* subscript. For applications where a single sub-flow is sufficient, such as the one discussed in Section 6, we propose to use a single network across all channels **(c)** to keep the evaluation cost largely agnostic to the channel count.

Modeling the CV shape with normalizing flows. Leveraging a normalizing flow to represent the shape $\bar{g}$ of the control variate is straightforward. We use the unit hypercube with the same number of dimensions D as g to be the latent space $\mathcal{L}$. The normalized CV is then modeled as

$$\bar{g}(x) := p_\mathcal{X}(x; \theta_{\bar{g}}) = p_\mathcal{L}(\widehat{x}) \cdot J(x; \theta_{\bar{g}}). \tag{7}$$

It is worth noting that the product on the right-hand side is normalized by construction: the probability density $p_\mathcal{L}$ is normalized by definition and each warp in the flow merely redistributes the density without altering the total mass. This is key for ensuring that $\bar{g}$ is and remains normalized during training.

In our implementation, the warps in the normalizing flow assume an autoregressive structure: dimension d in the output x_{i+1} of the i-th warp is conditioned on only the preceding dimensions in the input x_i:

$$x_{i+1}^d = h\big(x_i^d; m(x_i^{<d}; \phi_i^d)\big), \tag{8}$$

where the superscript $<d$ denotes the preceding dimensions, m is a neural network, and ϕ_i^d are its parameters, with $\theta_{\bar{g}} = \bigcup_i \bigcup_d \phi_i^d$. This ensures tractable Jacobian determinants that are computed as the product of diagonal terms in the Jacobian matrix of h. The diagonal terms are specific to the transform h being used—we use piecewise-quadratic warping functions proposed by Müller et al. [2019] in our implementation.

Figure 3(a) illustrates the autoregressive structure of the i-th warp in the normalizing flow. We adopt the terminology of Papamakarios et al. [2019] and refer to one autoregressive block as the "sub-flow". We utilize an independent network for inferring the warp of each dimension. The alternative of using a single network for all dimensions requires elaborate masking [Germain et al. 2015; Papamakarios et al. 2017] to enforce the autoregressive structure. Having an independent network per dimension simplifies the implementation and, more importantly, facilitates network sharing when dealing with multi-channel control variates.

Multi-channel CV. Many integration problems simultaneously operate on multiple, potentially correlated channels. In this article, for instance, we estimate spectrally resolved integrals; one for each RGB channel. In order to minimize the variance per channel, it is advantageous to use a separate control variate for each channel rather than sharing one CV across all channels.

The most straightforward solution is to instantiate a distinct normalizing flow for each channel; the per-channel sub-flows are illustrated in Figure 3(b); symbols were dropped for brevity. Unfortunately, this makes the computation cost linear in the number of channels—a penalty that we strive to avoid.

We propose to keep the cost largely constant by sharing corresponding neural networks across the channels. However, since network sharing introduces correlations across channels, e.g. red dimensions can influence green dimensions, special care must be taken to constrain the model correctly.

Merely concatenating the inputs to the k-th network across the per-channel flows, and instrumenting the network to produce parameters for warping dimensions in *all* n channels, is problematic as it corresponds to predicting a *single* normalized $(n \times D)$-dimensional function. Instead, we need n *individually* normalized, D-dimensional functions, like in the case of instantiating a distinct flow for each channel. We must ensure that each channel of the CV is normalized *individually*.

Note that since channels can influence each other only after the first sub-flow, the first sub-flow produces individually normalized functions, even if the networks are shared across the channels. This is easy to verify by inspecting the $nD \times nD$ Jacobian matrix constructed for all dimensions in all channels. The matrix will have a block-diagonal structure, where each $d \times d$ block corresponds to the Jacobian matrix of one of the channels. All entries outside of the blocks on the diagonal will be zero. This observation allows us to share the networks as long as we use only *one* sub-flow to model each channel of the CV shape; as illustrated in Figure 3(c). The benefits of sharing the networks are studied in Figure 4.

3.2 Modeling the Integral of the Control Variate

Representing the integral value by a neural network $G(y; \theta_G)$ is fairly straightforward as we can use any architecture. We exponentiate the network output to ensure that the CV integral is always positive. The combination of the exponentiated network output and the normalizing flow for the CV shape constrains the CV to be a non-negative function; negative values are excluded by design. This is desired for the light-transport application in Section 6 that deals with non-negative integrands only.

Note that even without the exponentiation, the neural CV may only be non-negative or non-positive. Then, signed integrands may be handled using the extension described in Section 8.1.

3.3 Modeling the CV Coefficient

Since the control variate may not match f perfectly—our neural CV is no exception—the variate is weighted by the CV coefficient α that controls its contribution. The optimal, variance-minimizing value of $\alpha(y)$ is known to be $\mathrm{Cov}(f(x, y), g(x, y))/\mathrm{Var}(g(x, y))$ [Lavenberg et al. 1982; Nelson 1990]. However, computing the optimal value, which generally varies with y, can be prohibitively expensive in practice. We thus model the coefficient using a neural network $\alpha(y; \theta_\alpha)$ with a sigmoid output activation that constrains its value to the interval $(0, 1)$ for numerical robustness. The network is trained to output the appropriate contribution of the CV in dependence on the parameter y. In Section 5, we contribute a loss function for optimizing the neural network $\alpha(y; \theta_\alpha)$ from Monte Carlo estimates such that it minimizes variance.

Since both α and G are mere scaling factors of g, one could model the product $\alpha \cdot G$ directly. We choose to keep them separate as this enables approximating F by (unweighted) G without evaluating the residual integral. We exercise this option in a biased version of our light-transport estimator whenever the approximation error is heuristically determined to be low; details follow in in Section 6.1.

4 MONTE CARLO INTEGRATION WITH NCV

As an evolution of the parametric integral in Equation (2), our parametric trainable control variate yields

$$F(y) = \alpha(y; \theta_\alpha) \cdot G(y; \theta_G)$$
$$+ \int_{\mathcal{D}} f(x, y) - \alpha(y; \theta_\alpha) \cdot g(x, y; \theta_g) \, dx, \qquad (9)$$

where the various θ denote the corresponding model parameter sets. For the sake of readability, we omit the dependency on y in the following derivations and define the shorthands $\hat{g}$ and $\hat{G}$ that represent the α-weighted CV and its corresponding integral:

$$\hat{G}(\theta_{\hat{G}}) := \alpha(\theta_\alpha) \cdot G(\theta_G); \qquad \theta_{\hat{G}} := \theta_\alpha \cup \theta_G, \qquad (10)$$
$$\hat{g}(x; \theta_{\hat{g}}) := \alpha(\theta_\alpha) \cdot g(x; \theta_g); \qquad \theta_{\hat{g}} := \theta_\alpha \cup \theta_G \cup \theta_{\hat{g}}. \qquad (11)$$

Applying these notational simplifications, a one-sample Monte Carlo estimator of Equation (9) amounts to

$$\langle F \rangle = \hat{G}(\theta_{\hat{G}}) + \frac{f(X) - \hat{g}(X; \theta_{\hat{g}})}{p(X; \theta_p)}, \qquad (12)$$

where $p(X; \theta_p) \equiv p(X, y; \theta_p)$ is the parametric probability density of drawing sample X.

Importance sampling of the residual integral. To motivate the need for a parametric PDF model, we note that the variance of Equation (12) is minimized when the PDF is proportional to the absolute correction term $|f(X) - \hat{g}(X; \theta_{\hat{g}})|$. Since the CV will be optimized progressively, the correction term will evolve over time. In the ideal case, the absolute difference $|f(X) - \hat{g}(X; \theta_{\hat{g}})|$ would get uniformly smaller and the optimal sampling distribution would be uniform, i.e. constant. However, our experiments showed that despite the approximation power of neural networks, the numerator is never sufficiently uniformly bounded to permit a uniform PDF $p_{\mathcal{U}}(x)$ to perform well in practice.

Accounting for the progressive optimization and the limited expressivity of the CV, we propose a sampling PDF that combines two samplers: a defensive sampler (in the following: uniform) that bootstraps the initial Monte Carlo estimates, and a learned parametric sampler that can capture the shape of the numerator once the CV has converged. We combine these two sampling distributions using multiple importance sampling (MIS) [Veach and Guibas 1995] with learned probabilities for selecting the PDFs [Müller et al. 2019].

We target the neural importance sampling probability density p_{NIS} [Müller et al. 2019] at the difference in the numerator of Equation (12). To probabilistically select between uniform sampling $p_{\mathcal{U}}$ and p_{NIS} we train a parametric neural network $c(x; \theta_c)$ that approximates the variance-optimal selection probabilities of p_{NIS}. We closely follow the approach by Müller et al. [2019] (including the prevention of degenerate training by the β parameter) optimizing $c(x; \theta_c)$ concurrently with the CV and PDF models to strike a good balance between uniform and neural importance sampling at any time during the training process. The final PDF reads:

$$p(x; \theta_p) = \left(1 - c(x; \theta_c)\right) p_{\mathcal{U}}(x) + c(x; \theta_c) \, p_{\mathrm{NIS}}(x; \theta_{\mathrm{NIS}}), \qquad (13)$$

where $\theta_p := \theta_c \cup \theta_{\mathrm{NIS}}$.

Spectral 2D example. We demonstrate the efficiency benefits of using our neural control variates for variance reduction in Figure 4. We compare neural importance sampling [Müller et al. 2019] alone to three flavors of our full estimator from Equation (12): (i) a monochromatic, single-channel flow, (ii) multiple independent flows (one per channel), and (iii) the proposed multi-channel flow. The multi-channel flow consistently achieves the highest efficiency, while learning only slightly worse control variates than multiple independent flows. Note how the sampling PDF focuses on the high-frequency detail that our control variates do not perfectly capture.

5 OPTIMIZATION

In this section, we derive a theoretically optimal, variance-minimizing loss for optimizing the parameters of the CV and the sampling distribution. We then propose an empirical, composite loss that provided better performance and stable optimization in our experiments.

5.1 Minimizing Variance by Optimization

Our goal is to minimize the variance of the CV estimator by training the neural networks using a convergent gradient-based optimizer. Stochastic gradient descent provably converges to local optima
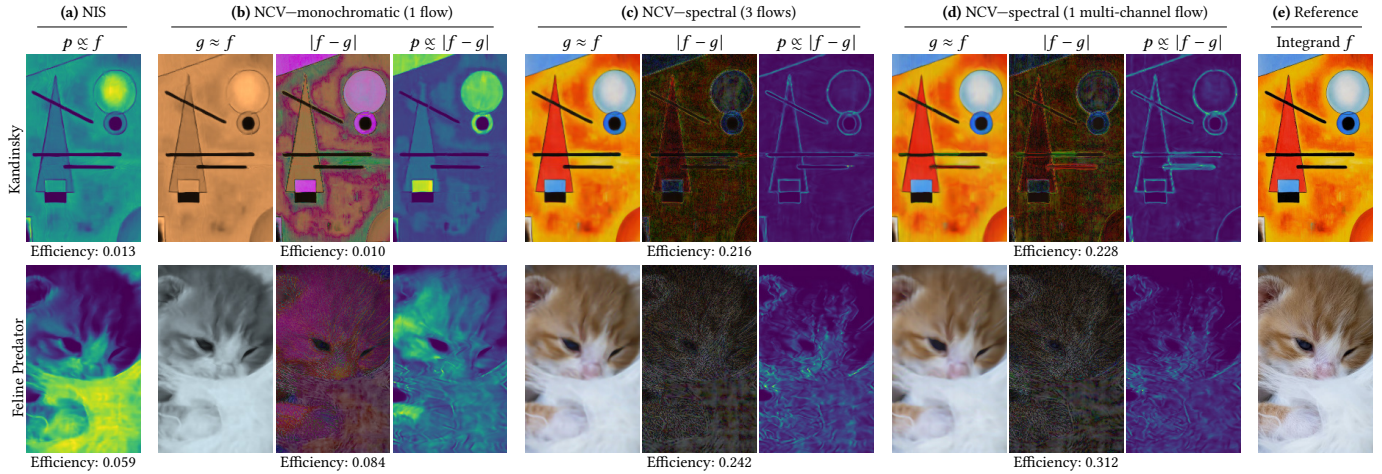
(a) NIS	**(b)** NCV—monochromatic (1 flow)	**(c)** NCV—spectral (3 flows)	**(d)** NCV—spectral (1 multi-channel flow)	**(e)** Reference
$p \propto f$	$g \approx f$ $\quad$ $|f-g|$ $\quad$ $p \propto |f-g|$	$g \approx f$ $\quad$ $|f-g|$ $\quad$ $p \propto |f-g|$	$g \approx f$ $\quad$ $|f-g|$ $\quad$ $p \propto |f-g|$	Integrand f

Kandinsky

Efficiency: 0.013 — Efficiency: 0.010 — Efficiency: 0.216 — Efficiency: 0.228

Feline Predator

Efficiency: 0.059 — Efficiency: 0.084 — Efficiency: 0.242 — Efficiency: 0.312

Fig. 4. Comparison of neural importance sampling **(a)** [Müller et al. 2019] and various flavors of our neural control variates **(b, c, d)** on two toy integration problems (rows). The integrands are 2D images with 3 color channels **(e)**; the goal is to estimate the average color of the images. We report the Monte Carlo efficiency, defined as $\left(\mathbb{V}[\langle F \rangle] \cdot \textbf{runtime}\right)^{-1}$, of using the different techniques after each technique's training converged. We also visualize the functions learned during the MC estimation, i.e. the sampling PDF p and the control variate g. NIS **(a)** is the least efficient method, because importance sampling can only target a scalar quantity—in this case the average of the 3 channels. Applying our NCVs, even with a single monochromatic flow **(b)**, improves efficiency, because the learned CV g is able to match the average color of the integrand. The learned sampling PDF p therefore only needs to focus on the remaining color variation in the residual difference $|f - g|$. Using three independent flows **(c)** and using one of our multi-channel flows **(d)** for the CV g both achieve great additional efficiency gains, because they can model color variation. The residual difference $|f - g|$ is thus much smaller and the sampling PDF p focuses on the remaining approximation error, which consists of sharp edges in the integrand. Our multi-channel flow yields the best MC efficiency: the fit is only slightly worse than with three flows but the evaluation is much faster.

when driven by *unbiased estimates* of the loss gradient.[1] In this section, we first derive the variance formula and then show that unbiased gradient estimates thereof can be computed using auto-differentiation.

We use the variance

$$\mathbb{V}[\langle F \rangle] = \mathbb{E}\left[\langle F \rangle^2\right] - \mathbb{E}[\langle F \rangle]^2$$

$$= \int_{\mathcal{D}} \frac{\left(f(x) - \hat{g}(x; \theta_{\hat{g}})\right)^2}{p(x; \theta_p)} \, dx - \left(F - \hat{G}(\theta_{\hat{G}})\right)^2 \quad (14)$$

of the estimator in Equation (12) as the loss function.

Interpretation. Minimizing the first term of Equation (14) corresponds to fitting $\hat{g}$ to f in terms of weighted least squares, where the weights are the inverse sampling density. The weighted-least-squares distance is minimized when $\hat{g}(x) = f(x)$, leading to zero variance. Interestingly, the variance is also zero when the non-zero first term equals to the second term. Due to this additional degree of freedom, there exists an entire family of CVs that yield zero variance. A classical example of such a configuration is a control variate that matches f up to an additive constant, $\hat{g}(x) = f(x) + c$ for $c \in \mathbb{R}$, and a uniform $p(x)$.

Variance with noisy estimates of $f(x)$. In many applications, the original integrand $f(x)$ cannot be evaluated *analytically*. One such

application is investigated in Section 6, where we apply control variates to light transport simulation governed by a Fredholm integral equation.

Generalizing Equation (14), we now demonstrate that noisy estimates of $f(x)$ pose no problem for the convergence of the optimizer. Using the generic notation $f(x) := \int_{\mathcal{P}} f(x, z) \, dz$ and inserting it into the integral in Equation (9) (with y being omitted for brevity as mentioned before), we obtain

$$F = \hat{G}(\theta_{\hat{G}}) + \int_{\mathcal{D}} \int_{\mathcal{P}} f(x, z) \, dz - \hat{g}(x; \theta_{\hat{g}}) \, dx . \quad (15)$$

A one-sample Monte Carlo estimator that leverages a single (X, Z) sample to approximate F reads

$$\langle F \rangle = \hat{G}(\theta_{\hat{G}}) + \frac{f(X, Z)}{p(X, Z; \theta_p)} - \frac{\hat{g}(X; \theta_{\hat{g}})}{p(X; \theta_p)}, \quad (16)$$

where $p(X, Z; \theta_p) = p(X; \theta_p) \cdot p(Z|X)$ is the joint probability density of sampling X and Z, and $p(X; \theta_p)$ and $p(Z|X)$ are the marginal and conditional densities, respectively.

The variance of the estimator in Equation (16) can be derived in analogy to the variance of the estimator in Equation (12):

$$\mathbb{V}[\langle F \rangle] = \int_{\mathcal{D}} \int_{\mathcal{P}} \left(\frac{f(x, z)}{p(z|x)} - \hat{g}(x; \theta_{\hat{g}})\right)^2 \frac{p(z|x)}{p(x; \theta_p)} \, dz dx$$

$$- \left(F - \hat{G}(\theta_{\hat{G}})\right)^2; \quad (17)$$

see Appendix A for a complete derivation.

Finding optimal $\theta_{\hat{g}}, \theta_{\hat{G}}, \theta_p$ that minimize Equation (17) in closed form is not practical as the equation contains the unknown integral F, which we are trying to compute in the first place, and a double

---

[1]For a formal proof of convergence, the learning rate must approach zero at a carefully chosen rate, leading to an impractically slow optimization. Leaving the learning rate high, the optimization fluctuates around local minima, which is a widely accepted limitation in machine learning literature.

integral, which for meaningful settings in computer graphics is infeasible to solve analytically. Therefore, we resort to stochastic gradient-based optimizers that converge to the correct solution even if the loss is only approximated; provided that its approximation is unbiased.

Taking advantage of autograd functionality. Using Leibniz's integral rule, we can swap the order of differentiation and MC estimation of variance: first estimate variance and then rely on auto-differentiation in modern optimization tools to compute the gradients. Using Monte Carlo, the variance in Equation (17) can be estimated using the following unbiased one-sample estimator:

$$\langle \mathbb{V}[\langle F \rangle] \rangle = \frac{\left(\frac{f(X,Z)}{p(Z|X)} - \hat{g}(X; \theta_{\hat{g}}) \right)^2 p(Z|X)}{p(X; \theta_p) \, q(X) \, q(Z|X)}$$
$$- \left(\frac{\langle f(X) \rangle}{q(X)} - \hat{G}(\theta_{\hat{G}}) \right)^2 , \tag{18}$$

where q is the density of samples used for estimating the variance. The estimator can be further simplified assuming that we use the same conditional densities in $\langle F \rangle$ and $\langle \mathbb{V} \rangle$, i.e. $p(z|x) = q(z|x)$, and interpreting the fraction $\frac{f(X,Z)}{p(Z|X)}$ as a one-sample estimator of $f(X)$:

$$\boxed{\begin{aligned} \langle \mathbb{V}[\langle F \rangle] \rangle &= \frac{\left(\langle f(X) \rangle - \alpha(\theta_\alpha) g(X; \theta_g) \right)^2}{p(X; \theta_p) \, q(X)} \\ &- \left(\frac{\langle f(X) \rangle}{q(X)} - \alpha(\theta_\alpha) G(\theta_G) \right)^2 , \end{aligned}} \tag{19}$$

where the symbols with hats were replaced by their definitions.

The variance estimate in Equation (19) can be used as the loss function in modern optimization tools based on autograd. Unfortunately, despite being theoretically optimal, our empirical analysis revealed poor overall performance and unstable optimization when using this loss.

5.2 Composite Loss for Stable Optimization

The variance of the parametric estimator, Equation (17), can be zero for an entire family of configurations of $\theta_\alpha, \theta_g, \theta_G$, and θ_p. However, taking into account the entire Equation (17) for each of the trainable components led to erratic optimization and often failed to approach one of the zero-variance configurations in our experiments.

We thus propose a composite loss that is more robust in the presence of noisy loss estimates. Our composite loss imposes restrictions as it is zero *only* for the following zero-variance configuration:

$$G(\theta_G) = F, \tag{20}$$

$$\bar{g}(x; \theta_{\bar{g}}) = \frac{f(x)}{F}, \tag{21}$$

$$p(x; \theta_p) = \frac{|f(x) - g(x; \theta_g)|}{\int_{\mathcal{D}} |f(x) - g(x; \theta_g)| \, \mathrm{d}x}, \text{ and} \tag{22}$$

$$\alpha(\theta_\alpha) = 1. \tag{23}$$

Despite being more restrictive, decomposing the optimization into smaller, better-understood optimization tasks leads to better results in practice than blindly relying on Equation (17). Our composite

loss is the sum of the individual terms:

$$\mathcal{L} = \underbrace{\mathcal{L}^2(F, G; \theta_G)}_{\text{CV integral}} + \underbrace{\mathcal{L}_H(\bar{f}, \bar{g}; \theta_{\bar{g}})}_{\text{CV shape}} + \underbrace{\mathcal{L}_H(|f - g|, p; \theta_p)}_{\text{Sampling PDF}} + \underbrace{\mathcal{L}_{\mathbb{V}}(\theta_\alpha)}_{\alpha\text{-coefficient}}, \tag{24}$$

which we detail in the following paragraphs.

CV integral optimization. To satisfy the constraint in Equation (20), we minimize a relative $\mathcal{L}^2$ metric

$$\mathcal{L}^2(F, G; \theta_G) = \frac{\left(F - G(\theta_G) \right)^2}{\text{sg}(G(\theta_G))^2 + \epsilon}, \tag{25}$$

where $\text{sg}(x)$ indicates that x is treated as a constant, i.e. no gradients w.r.t. it are computed, and $\epsilon = 1 \times 10^{-2}$. Our choice of a relative $\mathcal{L}^2$ metric has two reasons: first, the $\mathcal{L}^2$ metric admits unbiased gradient estimates when F is noisy, and second, relative losses are robust with respect to a high dynamic range of values. We use $G(\theta_G)^2$ as the normalization constant, as proposed by Lehtinen et al. [2018], because normalizing by F^2 [Rousselle et al. 2011] is infeasible—our goal is to estimate F in the first place. $G(\theta_G)^2$ merely serving as an approximation of F^2 in the denominator is the reason why it must be treated as a constant for the optimization to be correct—hence the $\text{sg}(\cdot)$ around it. It follows that our Monte Carlo estimator of $\mathcal{L}^2(F, G; \theta_G)$, which we feed to automatic differentiation, reads

$$\langle \mathcal{L}^2(F, G; \theta_G) \rangle = \frac{\left(\langle F \rangle - G(\theta_G) \right)^2}{\text{sg}(G(\theta_G))^2 + \epsilon}. \tag{26}$$

In Figure 5, we illustrate the learned integral when optimizing either the variance, $\mathcal{L}^2$, or relative $\mathcal{L}^2$ in the setting of light-transport simulation as explored in Section 6. The relative $\mathcal{L}^2$ loss achieves the most accurate fit.

CV shape optimization. The CV shape is modeled using a normalizing flow, the parameters of which are optimized using the cross entropy. The cross entropy measures the similarity between two normalized functions and yields more robust convergence than minimizing variance directly [Müller et al. 2019]. Since we aim to satisfy the constraint in Equation (21), we minimize the cross entropy of the normalized integrand, $\bar{f}(x) = f(x)/F$, to the shape of the CV, $\bar{g}$:

$$\mathcal{L}_H(\bar{f}, \bar{g}; \theta_{\bar{g}}) = - \int_{\mathcal{D}} \bar{f}(x) \log(\bar{g}(x; \theta_{\bar{g}})) \, \mathrm{d}x. \tag{27}$$

The main caveat of the cross entropy is that it requires normalizing the integrand. Despite F not being known exactly, approximate normalization $\bar{f}(x) \approx f(x)/G(\theta_G)$ is feasible by using the learned CV integral $G(\theta_G)$ instead of the exact normalization constant. With this observation, an approximately normalized MC estimator of the cross entropy that can be fed to automatic differentiation reads

$$\langle \mathcal{L}_H(\bar{f}, \bar{g}; \theta_{\bar{g}}) \rangle = - \frac{\langle f(X) \rangle}{\text{sg}(G(\theta_G)) + \epsilon} \frac{\log(\bar{g}(X; \theta_{\bar{g}}))}{q(X)}. \tag{28}$$

Sampling distribution optimization. Our parametric sampling distribution $p(x; \theta_p)$ consists of a normalizing flow $p_{\text{NIS}}(x; \theta_{\text{NIS}})$ as well as the selection probability $c(x; \theta_c)$, which are both optimized using the cross entropy—the same as in neural importance sampling (NIS) [Müller et al. 2019]. However, in contrast to NIS, which optimizes the flow to match the normalized integrand $\bar{f}(x)$, we optimize
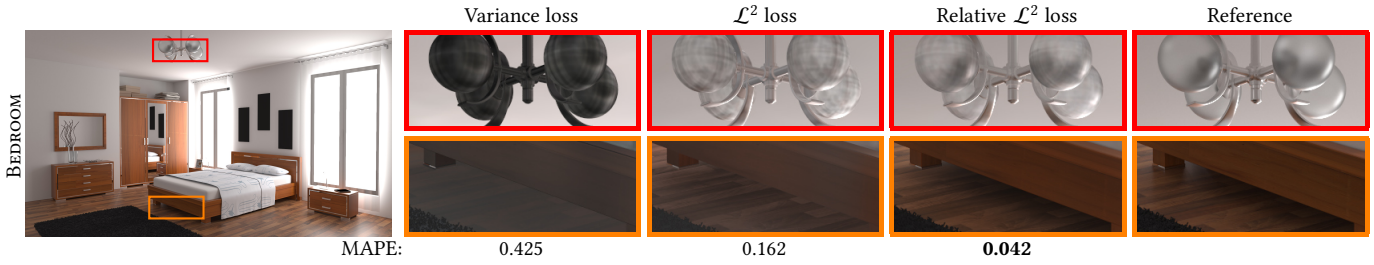
	Variance loss	$\mathcal{L}^2$ loss	Relative $\mathcal{L}^2$ loss	Reference
MAPE:	0.425	0.162	**0.042**	

Fig. 5. Control Variate integral optimization. Using the estimator variance loss (left) to optimize the CV integral G effectively merges it with the coefficient α, resulting in a darker output where the CV shape is a poor match for the target. Using the $\mathcal{L}^2$ loss (middle) decouples the CV integral and the coefficient α. The relative $\mathcal{L}^2$ loss (right) further improves the model prediction in dark regions, such as the floor under the bed.

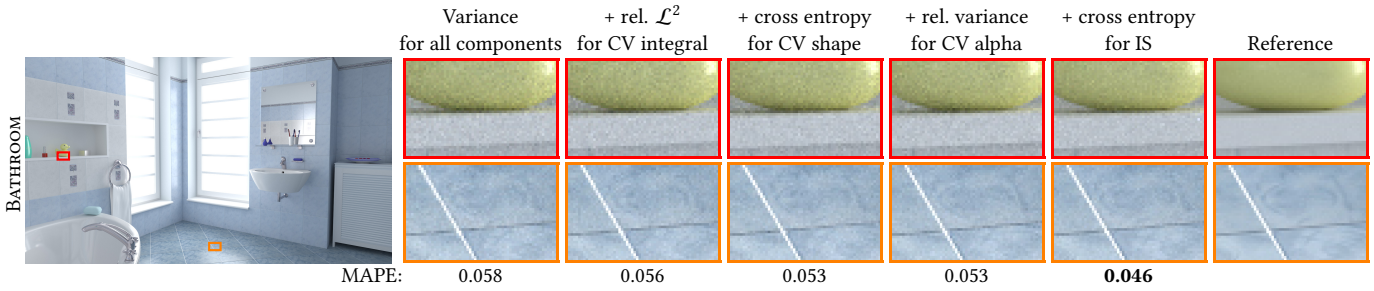|	Variance for all components	+ rel. $\mathcal{L}^2$ for CV integral	+ cross entropy for CV shape	+ rel. variance for CV alpha	+ cross entropy for IS	Reference
MAPE:	0.058	0.056	0.053	0.053	**0.046**	

Fig. 6. Impact of the individual composite-loss terms at equal sample count (512 spp). First, we jointly optimize all components using the variance loss described in Section 5.1. Then, we progressively replace each variance term with the corresponding term from the composite loss described in Section 5.2. Optimizing the CV integral using the relative $\mathcal{L}^2$ loss has a minor impact on the final reconstruction, but it greatly improves the visual accuracy of the CV integral (as shown in Figure 5), which in turn enables its use in our biased reconstruction. The accurate CV integral is also a key enabler of the cross entropy loss that we use to optimize the CV shape and IS component, which proved crucial for the overall robustness and efficiency of our algorithm. Aside of the numerical and visual improvements, the main benefit of our composite loss lies in its robustness: rendering the Bathroom scene using our proposed composite loss consistently converged to a satisfying result, whereas using the variance loss frequently resulted in diverging optimizations.

the flow to approximate the normalized absolute difference in Equation (22). Once again, we approximate the normalization constant by $G(\theta_G)$. In addition, we approximate the difference $|f(x) - g(x;\theta_g)|$ using the biased estimator

$$\Delta_{f,g}(X) = |\langle f(X) \rangle - \mathrm{sg}(g(X;\theta_g))|,$$

resulting in the following cross-entropy estimator for automatic differentiation:

$$\langle \mathcal{L}_H(|f - g|, p; \theta_p) \rangle = -\frac{\Delta_{f,g}(X)}{\mathrm{sg}(G(\theta_G)) + \epsilon} \frac{\log(p(X; \theta_p))}{q(X)}. \quad (29)$$

Note that $\Delta_{f,g}(X)$ is biased due to Jensen's inequality: taking the absolute value of an estimator *overestimates* the absolute value of the estimator's expectation. As a result, the above cross-entropy estimator is an *upper bound* to the true cross entropy between $|f - g|$ and p. Crucially, since the upper bound has the same minimum as the cross entropy (when the flow matches the normalized absolute difference) minimizing the upper bound does not prevent convergence and worked sufficiently well in our experiments.

α-coefficient optimization. As given by the constraint in Equation (23), we only achieve zero variance using $\alpha = 1$. However, this identity assumes that our parametric control variate and sampling distribution *exactly* match their targets, which is unlikely in practice. In such cases, the α-coefficient allows for downweighting the

control variate to avoid increased variance due to a poor fit. We therefore employ a parametric model for α, too, and optimize it to minimize the *relative* variance of the complete CV estimator:

$$\mathcal{L}_{\mathbb{V}}(\theta_\alpha) = \frac{\mathbb{V}[\langle F \rangle]}{\mathrm{sg}(G(\theta_G))^2 + \epsilon}, \quad (30)$$

where we use a relative loss for the same reason as in Equation (25): to be robust with respect to a high dynamic range of values. The α coefficient is thus the only component of our model that is optimized with respect to the variance loss in Equation (17); we use the estimator in Equation (19) to estimate the numerator of $\mathcal{L}_{\mathbb{V}}(\theta_\alpha)$ for optimizing θ_α:

$$\langle \mathcal{L}_{\mathbb{V}}(\theta_\alpha) \rangle = \frac{\langle \mathbb{V}[\langle F \rangle] \rangle}{\mathrm{sg}(G(\theta_G))^2 + \epsilon}, \quad (31)$$

In Figure 6, we demonstrate the additional robustness of using our composite loss instead of the theoretically optimal variance loss. We note that the variance loss result, as well as the intermediate ablation results, are generally very unstable; we had to run many optimization runs to produce these results. In contrast, the final composite loss consistently produces useful results.
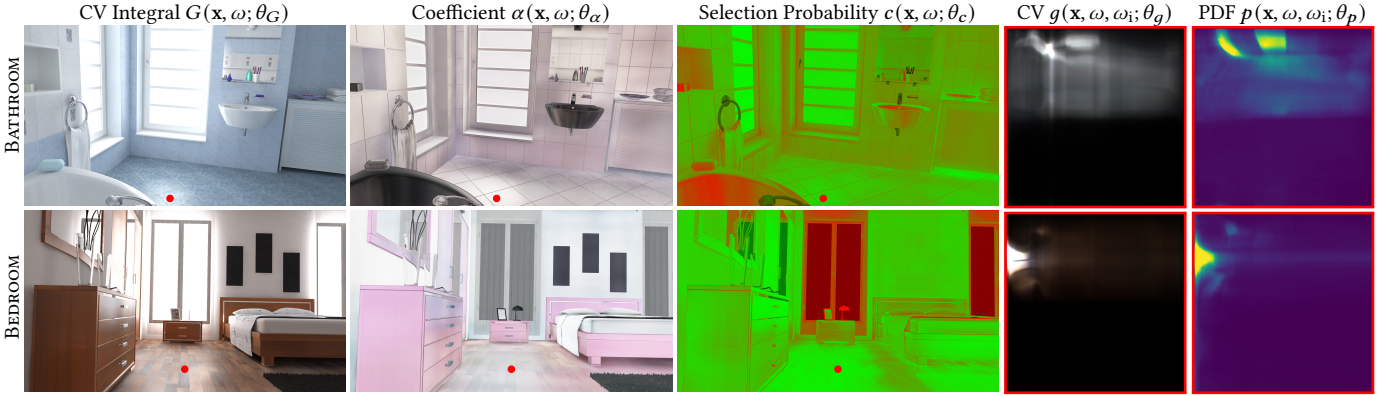
Fig. 7. All learned components of our method when applied to light transport simulation: we visualize the learned CV integral $G(\mathbf{x}, \omega; \theta_G)$, the coefficient $\alpha(\mathbf{x}, \omega; \theta_\alpha)$, and selection probability $c(\mathbf{x}, \omega; \theta_c)$ at the primary vertices (first non-delta interaction) of each pixel. Furthermore, we show the directionally resolved learned CV $g(\mathbf{x}, \omega, \omega_i; \theta_{\bar{g}})$ and PDF $p(\mathbf{x}, \omega, \omega_i; \theta_p)$ at the spatial location marked in red. The CV integral approximates the scattered light field $L_s(\mathbf{x}, \omega)$ remarkably well. In places where either the CV integral or the shape is inaccurate, the learned alpha-coefficient weighs down the contribution of the CV to our unbiased estimator. Lastly, the learned selection probability blends between BSDF sampling (red) and residual neural importance sampling (green) such that variance is minimized. Note how glossy surfaces tend to favor BSDF sampling, whereas rougher surfaces often favor residual NIS.

6 APPLICATION TO LIGHT TRANSPORT

With trainable control variates at hand, we are ready to demonstrate their benefits in light-transport simulation. Physically based image synthesis is concerned with estimating the scattered radiance

$$L_s(\mathbf{x}, \omega) = \int_{S^2} f_s(\mathbf{x}, \omega, \omega_i) L_i(\mathbf{x}, \omega_i) |\cos \gamma| \, d\omega_i \qquad (32)$$

that leaves surface point $\mathbf{x}$ in direction ω [Pharr et al. 2016], where f_s is the bidirectional scattering distribution function, L_i is radiance arriving at $\mathbf{x}$ from direction ω_i, and γ is the foreshortening angle.

The correspondence to Equation (9) is established as follows: the scattered radiance $L_s(\mathbf{x}, \omega)$ corresponds to the parametric integral $F(y)$, where $y \equiv (\mathbf{x}, \omega)$, which we refer to as the *query location*. The integration domain and the integration variable are the unit sphere and the direction of incidence, i.e. $\mathcal{D} \equiv S^2$ and $x \equiv \omega_i$, respectively.

Our goal is to reduce estimation variance by leveraging the parametric CV from Section 3. Its integral component serves as an approximation of the scattered radiance, i.e. $G(\mathbf{x}, \omega; \theta_G) \approx L_s(\mathbf{x}, \omega)$, while its shape component $\bar{g}(\mathbf{x}, \omega, \omega_i; \theta_{\bar{g}})$ approximates the normalized integrand. In analogy to Equation (12), a one-sample MC estimator of Equation (32) with the trainable CV from Section 3 reads:

$$\langle L_s(\mathbf{x}, \omega) \rangle = \hat{G}(\mathbf{x}, \omega; \theta_G)$$
$$+ \frac{f_s(\mathbf{x}, \omega, \Omega) L_i(\mathbf{x}, \Omega) |\cos \gamma| - \hat{g}(\mathbf{x}, \omega, \Omega; \theta_{\bar{g}})}{p(\Omega | \mathbf{x}, \omega; \theta_p)} . \quad (33)$$

We made one small modification to $p(\Omega | \mathbf{x}, \omega; \theta_p)$: instead of mixing NIS with uniform sampling as proposed in Section 4, we mix NIS with BSDF sampling p_{f_s}, which in rendering in many cases is a better baseline than uniform sampling. This results in the following PDF:

$$p(\Omega | \mathbf{x}, \omega; \theta_p) = (1 - c(\mathbf{x}, \omega; \theta_c)) \, p_{f_s}(\Omega | \mathbf{x}, \omega)$$
$$+ c(\mathbf{x}, \omega; \theta_c) \, p_{NIS}(\Omega | \mathbf{x}, \omega; \theta_{NIS}) . \qquad (34)$$

Figure 7 visualizes how each component of our trainable CVs fits into the light-transport integral equation.

6.1 Path Termination

The recursive estimation of radiance terminates when the path escapes the scene or hits a black-body radiator that does not scatter light. Since the integral component of the CV approximates the scattered light field well in many cases, we considered skipping the evaluation of the correction term, thereby truncating the path and producing a biased radiance estimate. Figure 8 (column CV Integral) visualizes the neural scattered light field G at non-specular surfaces that are directly visible from the camera or seen through specular interactions. Compared to the reference (right-most column) the approximation error of the neural light field, which manifests as low-frequency variations and blurry appearance, is not suitable for direct visualization. However, deferring the approximation error to higher-order bounces (such as in final gathering for photon mapping) may strike a good balance between visual quality and computation cost ("Biased NCV (Ours)" column in Figure 8).

We utilize a simple criterion for ignoring the correction term. The criterion measures the stochastic area-spread of path vertices, which [Bekaert et al. 2003] proposed to use as the photon-mapping filter radius. Once the area spread becomes sufficiently large, we terminate the path and approximate $L_s(\mathbf{x}, \omega)$ by $G(\mathbf{x}, \omega; \theta_G)$.

Sampling of direction ω using $p(\Omega | \mathbf{x}, \omega)$ at path vertex $\mathbf{x}$ induces the area spread of

$$a(\mathbf{x}', \mathbf{x}) = \frac{1}{p(\mathbf{x}' | \mathbf{x}, \omega)} = \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{p(\Omega | \mathbf{x}, \omega) \, |\cos \gamma'|} \qquad (35)$$

around the next path vertex $\mathbf{x}'$, where γ' is the incidence angle at $\mathbf{x}'$. The cumulative area spread at the n-th path vertex is the convolution of the spreads induced at all previous vertices. Assuming isotropic Gaussian spreads with variance $\sqrt{a(\mathbf{x}', \mathbf{x})}$ and parallel surfaces, this

convolution can be approximated as:

$$a(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \left(\sum_{i=2}^{n} \sqrt{a(\mathbf{x}_i, \mathbf{x}_{i-1})} \right)^2 . \qquad (36)$$

We compare this cumulative area spread to the pixel footprint projected onto the primary vertex $\mathbf{x}_1$. If the projected pixel footprint is more than $10\,000\times$ smaller than the path's cumulative area spread—loosely corresponding to a 100-pixel-wide image-space filter—we terminate the path into $G(\mathbf{x}, \omega_i)$. Otherwise, we keep applying our unbiased control variates and recursively evaluate the heuristic at the next path vertex.
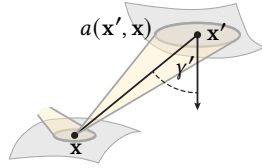
The heuristic path termination shortens the mean path length and removes a significant amount of noise at the cost of a small amount of visible bias; see Figure 1 and Section 7.

Our heuristic area spread is a simplified version of path differentials and could be made more accurate by taking into account anisotropy and additional dimensions of variation, for instance via covariance tracing [Belcour et al. 2013]. In our experiments, our heuristic worked sufficiently fine and hence we leave this extension to future work. Stochastic path termination via Russian roulette is discussed in Section 8.

6.2 Implementation

We implemented our neural control variates as well as neural importance sampling within Tensorflow [Abadi et al. 2015]. The rendering algorithm is implemented in the Mitsuba renderer [Jakob 2010], interfacing with Tensorflow to invoke the neural networks.

Rendering and training happen *simultaneously*, following the methodology of Neural Importance Sampling [Müller et al. 2019]: we begin by initializing the trainable parameters using Xavier initialization [Glorot and Bengio 2010] and then optimize our composite loss (Equation (24)) using Adam [Kingma and Ba 2014]. We use the CPU to perform light-transport computations and two GPUs to perform the neural-network-related computations. One GPU is responsible solely for training whereas the other is responsible for utilizing the current trained model to reduce variance as per Equation (33). Training and variance reduction mutually benefit each other—the models are synced every second—making our algorithm a variant of reinforcement learning.

Mitsuba communicates with Tensorflow in batches of 65 536 samples, where every *path vertex* is a single sample. At each path vertex $(\mathbf{x}, \omega)$, we initially proceed identically to NIS [Müller et al. 2019]: the renderer first queries the MIS selection probabilities $c(\mathbf{x}, \omega; \theta_{f_s})$ and $c(\mathbf{x}, \omega; \theta_{\mathrm{NIS}})$. Next, according to the selection probabilities, the renderer probabilistically selects either BSDF sampling or NIS. If BSDF sampling is selected, the algorithm queries the NIS PDF for the sampled direction ω_i. If NIS is selected, the algorithm queries a sample of ω_i via NIS. The renderer continues by querying the CV integral $G(\mathbf{x}, \omega)$, the CV shape $g(\mathbf{x}, \omega, \omega_i)$, as well as $\alpha(\mathbf{x}, \omega)$, and applies them according to Equation (33). After a light path has been completed, the reflected radiance at each vertex, along with the

Table 1. Parameters y that are fed to our parametric models along with their encoding and dimensionality. We apply one-blob (ob) encoding [Müller et al. 2019] to all parameters except for the reflectances and the transmittance.

Parameter	Symbol	Encoding
Scattered dir.	$\omega \in S^2$	$\mathrm{ob}(\omega/2 + 0.5) \in \mathbb{R}^{3 \times 32}$
Position	$\mathbf{x} \in \mathbb{R}^3$	$\mathrm{ob}(\mathbf{x}) \in \mathbb{R}^{3 \times 32}$
Path length	$k \in \mathbb{N}$	$\mathrm{ob}(k/k_{\max}) \in \mathbb{R}^{32}$
Surface normal	$\vec{n}(\mathbf{x}) \in S^2$	$\mathrm{ob}(\vec{n}(\mathbf{x})/2 + 0.5) \in \mathbb{R}^{3 \times 32}$
Surface roughness	$r(\mathbf{x}, \omega) \in \mathbb{R}$	$\mathrm{ob}\left(1 - e^{-r(\mathbf{x}, \omega)}\right) \in \mathbb{R}^{32}$
Diffuse reflectance	$f_{\mathrm{dr}}(\mathbf{x}, \omega) \in \mathbb{R}^3$	$f_{\mathrm{dr}}(\mathbf{x}, \omega) \in \mathbb{R}^3$
Specular reflectance	$f_{\mathrm{sr}}(\mathbf{x}, \omega) \in \mathbb{R}^3$	$f_{\mathrm{sr}}(\mathbf{x}, \omega) \in \mathbb{R}^3$
Transmittance	$f_{\mathrm{t}}(\mathbf{x}, \omega) \in \mathbb{R}^3$	$f_{\mathrm{t}}(\mathbf{x}, \omega) \in \mathbb{R}^3$

vertex's metadata, is put into a ring buffer that keeps track of the past 1 048 576 vertices. The training GPU continuously assembles training batches by picking random samples from the ring buffer to minimize correlations within each batch.

Specular BSDFs. BSDFs with Dirac-delta components (henceforth referred to as "specular") typically require special treatment because they are not square integrable. Inserting specular components into our equations results in the following behavior that needs to be explicitly implemented. There are two cases: (i) the BSDF has specular *and* non-specular components. In this case, the selection probability c is used in the standard way to select either BSDF sampling or NIS. If BSDF sampling is selected and one of its specular components is sampled, then the NIS PDF and our parametric control variate are treated as zero. Otherwise (i.e. when either NIS or a smooth BSDF component is sampled), we apply the neural control variate, but with g multiplied by the total probability of sampling NIS or a smooth BSDF component; G should not be multiplied by this number. (ii) the BSDF has only specular components. In this case, regular path tracing is used (without the influence of any of our parametric models).

Iterative rendering. We apply the same iterative rendering scheme as Müller et al. [2019]: we render $M = \lfloor \log_2(N + 1) \rfloor$ images with power-of-two sample counts $2^i; i \in \{0, \ldots, M\}$, except for the last iteration which may have fewer samples due to running out of render time. To obtain the final image, we average all images, weighted by the reciprocal of a robust numerical estimate of their mean pixel variance [Müller 2019] in order to limit the impact of high-variance initial samples.

Parameter augmentation for neural networks. As observed by Ren et al. [2013], the approximation power of a parametric model to learn the light field as a function of $(\mathbf{x}, \omega)$ may be dramatically improved when additional quantities are provided as input. Table 1 lists all parameters that we feed to our parametric models in addition to the query location and direction $(\mathbf{x}, \omega)$: the surface normal, the surface roughness, the diffuse and specular reflectance, and the transmittance. Directions are parameterized in a global coordinate frame as done by Müller et al. [2017].

We also include the path length k when the maximum path length is capped to some finite number $k_{\max}$; in all our results we use $k_{\max} = 10$. In this case, the networks must learn progressively less indirect illumination as k approaches $k_{\max}$.

All quantities are normalized such that they fall within the unit hypercube of their respective dimensionality. Those quantities that have a highly non-linear relationship with the light field (all but the reflectances and the transmittance) are additionally one-blob encoded [Müller et al. 2019], denoted by $ob(x)$.

Network and flow architecture. For both normalizing flows, i.e. the multi-channel flow for the CV shape and the standard flow for NIS, we use the piecewise-quadratic warp proposed by [Müller et al. 2019] with 64 bins and a uniform latent distribution $p_{\mathcal{L}}(x') \equiv p_{\mathcal{U}}(x')$. Both flows use $L = 2$ warps to make the total number of warps, neural networks, and trainable parameters comparable to standalone NIS [Müller et al. 2019], which uses a single flow with $L = 4$ warps.

All neural networks—i.e. those that parameterize our warps as well as the one that predicts G, α, and c—use the same architecture: a fully connected residual network [He et al. 2016] with 2 residual blocks that each have 2 layers with 256 neurons. We also experimented with other architectures, such as multi-layer perceptrons (MLPs) and U-nets (as proposed for NIS), and remark that architectural differences had only small, almost immeasurable impact on the results in our tests.

Lastly, matrix multiplications are computed at half precision in order to take advantage of dedicated hardware.

Optimization. We optimize the neural networks *during* rendering in a reinforcement-learning fashion: the vertices of traced paths are used to optimize the networks by minimizing Equation (24), while *simultaneously* the current neural networks are used to drive variance reduction via Equation (33). The networks reduce variance of their own training data *and* that of the final image.

We minimize Equation (24) using Adam [Kingma and Ba 2014] and use a learning rate of 1×10^{-3}, which decays in two steps: (i) $\sqrt{10} \times 10^{-4}$ after 25% of the rendering process and (ii) 1×10^{-4} after 50% of the rendering process. This learning-rate decay addresses a problem pointed out by Müller et al. [2019], where learned distributions exhibited prolonged fluctuations in the later stages of training.

Lastly, we note that we do *not* use batch normalization [Ioffe and Szegedy 2015], because it detrimentally affected computational *and* qualitative performance.

7 RESULTS AND ANALYSIS

All results were produced on an NVIDIA DGX-1, using one Intel Xeon E5–2698 v4 CPU (20 cores; 40 threads) and two Tesla V100 GPUs (comparable to two RTX 2080Ti). To gauge the practical usefulness of our technique, we compare render quality at *equal time*, but we recognize that the performance depends strongly on the particular hardware setup. Therefore, we also report *samples per pixel* in Table 2 for completeness.

We quantify rendering error using the "mean absolute percentage error" (MAPE), which strikes a good balance between being perceptually accurate and correlating with Monte Carlo standard deviation. MAPE is defined as $\frac{1}{N} \sum_{i=1}^{N} |v_i - \hat{v}_i|/(\hat{v}_i + \epsilon)$, where $\hat{v}_i$ is

the value of the i-th pixel in the reference image, v_i is the value of the i-th rendered pixel, and $\epsilon = 0.01$ prevents near-black pixels from dominating the metric. A rough estimate of Monte Carlo efficiency can be obtained by the *reciprocal square root* of MAPE—i.e. a 2× smaller MAPE loosely corresponds to 4× faster rendering.

Table 2 and Figure 8 summarize our main results. We report MAPE and samples per pixel after 2 hours of rendering at a resolution of 1920x1080. We compare unidirectional path tracing (PT), practical path guiding with recent improvements (PPG) [Müller 2019; Müller et al. 2017], neural importance sampling [Müller et al. 2019] without (NIS) and with (NIS++) our additionally proposed learning-rate decay and approximately normalized cross-entropy loss (Section 5.2, Equation (29)), and our neural control variates (NCV). Russian roulette is *not* used, which we elaborate on in Section 8. Among these unbiased techniques, our NCVs usually yield the lowest error.

Comparison to NIS and NIS++. To rule out the possibility of NCVs outperforming NIS simply because it uses additional neural networks for the CV components, we use only $L = 2$ warps for importance sampling the residual integral (NIS uses $L = 4$ warps).

Furthermore, we use a *single* neural network to simultaneously predict the coefficient α, the CV integral G, and the selection probability c. Our NCVs therefore use the same total number of neural networks (five), all with the same architecture, and the same total number of piecewise-quadratic warps (four) as NIS; therefore the number of trainable parameters is the same and the performance is comparable. Differences in the number of per-pixel samples are largely due to differences in importance sampling and therefore different mean path lengths.

Path termination using NCV. We also show the results of applying our path termination heuristic (Section 6.1) as a by-product of NCVs. The technique dramatically outperforms the unbiased algorithm at the cost of minimally visible artifacts (cf. the "NCV + heuristic" column in Figure 8). Please refer to the supplementary material with an interactive image viewer for full-resolution images.

Quality of approximations. Table 2 and Figure 8 include a column listing the MAPE obtained when evaluating the learned CV integral at the first non-specular path vertex. Even though the CV integral exhibits visible bias, its relatively low MAPE is an indicator of the excellent approximation power of neural networks.

To further explore the limits of our parametric models, we visualize them *from novel viewpoints* in Figure 10. For each scene in the figure, the neural networks were trained while rendering the corresponding entries in Table 2. In the BEDROOM and BATHROOM scenes, the specular highlights on the floor and the furniture are at the correct positions. This observation supports the claim that the neural networks learn the actual 5D light field, as opposed to a mere screen-space approximation of it. On the other hand, in the SPACESHIP scene, the highly glossy transport is not accurately captured from the novel viewpoint, despite the good performance of our NCVs in terms of MAPE. We show animated camera trajectories in the supplementary video, which also features the NECKLACE scene as another failure case of incorrectly learned glossy light transport.
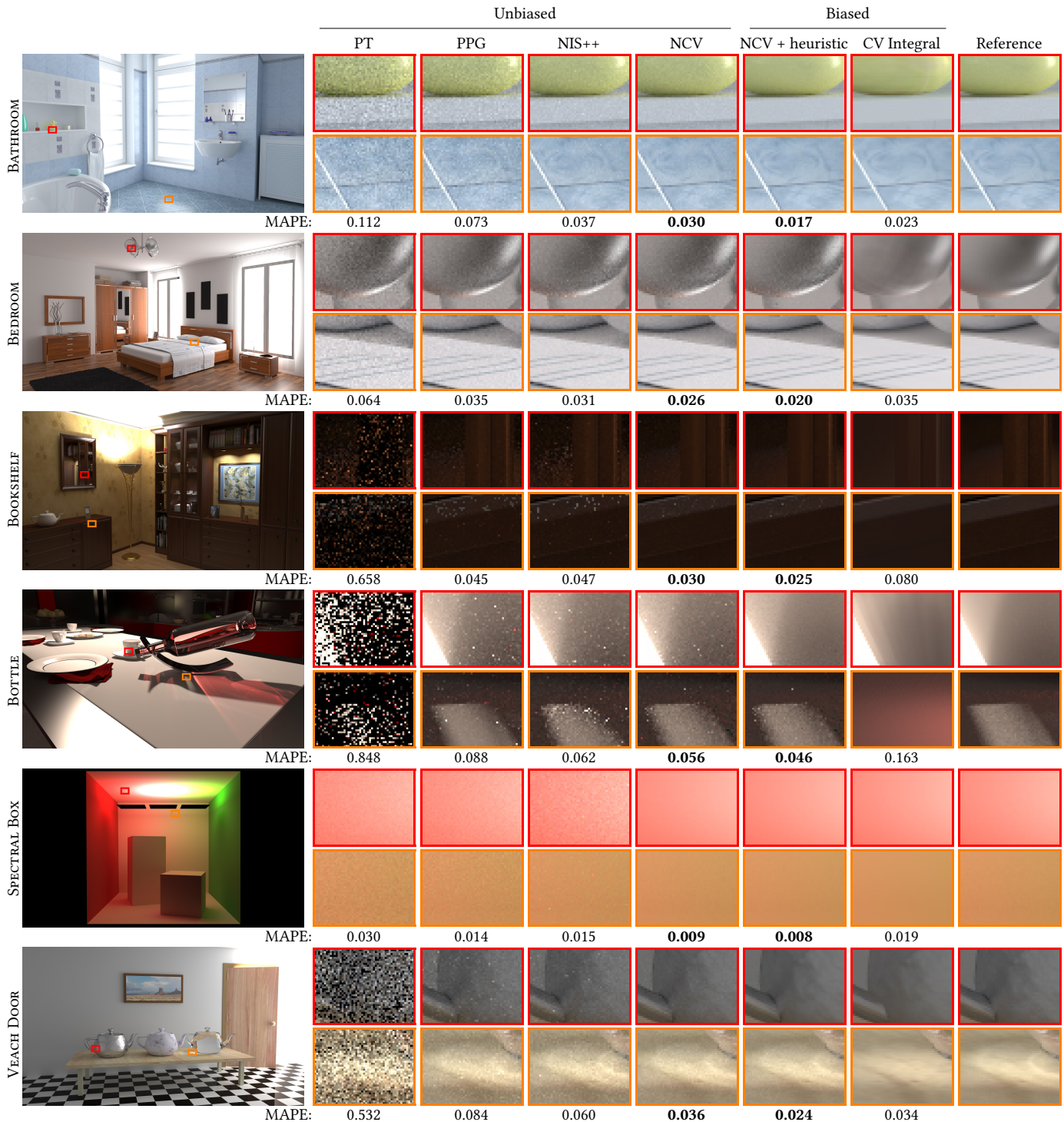
	Unbiased				Biased		
	PT	PPG	NIS++	NCV	NCV + heuristic	CV Integral	Reference
BATHROOM MAPE:	0.112	0.073	0.037	**0.030**	**0.017**	0.023	
BEDROOM MAPE:	0.064	0.035	0.031	**0.026**	**0.020**	0.035	
BOOKSHELF MAPE:	0.658	0.045	0.047	**0.030**	**0.025**	0.080	
BOTTLE MAPE:	0.848	0.088	0.062	**0.056**	**0.046**	0.163	
SPECTRAL BOX MAPE:	0.030	0.014	0.015	**0.009**	**0.008**	0.019	
VEACH DOOR MAPE:	0.532	0.084	0.060	**0.036**	**0.024**	0.034	

Fig. 8. Neural control variates (NCV) compared to our improved variant (NIS++) of neural importance sampling [Müller et al. 2019], practical path guiding with recent improvements (PPG) [Müller 2019], and uni-directional path tracing (PT). All images have a resolution of 1920 × 1080 pixels and were rendered in 2 hours. Unbiased NCV consistently achieves a moderate MAPE improvement. The biggest benefit is seen in scenes with smooth, indirect illumination (BATHROOM, SPECTRAL BOX, and VEACH DOOR). Approximating the tail contribution of light-transport paths by the learned CV integral driven by our heuristic described in Section 6.1 ("NCV + heuristic" column) allows for additional improvements, most noticeable in the BATHROOM and VEACH DOOR scenes. This variant of NCV reduces noise significantly while introducing only little visible bias, unlike naïvely evaluating the CV integral at the first non-specular path vertex ("CV Integral" column).

Table 2. We report equal-time *mean absolute percentage error* (MAPE) of several machine-learning-based variance reduction techniques on 18 test scenes. Bold entries indicate lowest error among unbiased/biased techniques. All images have a resolution of 1920 × 1080 pixels and were rendered in 2 hours. The achieved samples per pixel are written next to the error numbers. Our unbiased neural control variates (NCV) outperform neural importance sampling (NIS and our improved NIS++) on almost all scenes, except for the Sponza Atrium. The performance advantage is larger when the illumination is mostly indirect, such as in interior scenes (e.g. Bathroom, Spectral Box, Bookshelf and Veach Door). Using the integral G of the neural control variate without correcting for its error after interactions with rough surfaces (NCV + heuristic) results in a biased image, but in most cases with a significant additional error reduction—this implies that the variance reduction outweighs the amount of introduced bias. In one scene (Veach Lamp), lowest MAPE is achieved by using uncorrected G already at the first camera vertex (CV Integral), but the produced image suffers from visually displeasing artifacts (see Figure 8 and the supplementary material).

|	Unbiased					Biased	
	[Müller 2019]	[Müller et al. 2019]		Ours			
	PT	PPG	NIS	NIS++	NCV	NCV + heuristic	CV Integral
Artroom	1.393 1,756spp	0.108 2,998spp	0.079 1,231spp	0.062 1,271spp	**0.056** 1,188spp	**0.044** 1,188spp	0.091 1,188spp
Bathroom	0.112 1,785spp	0.073 2,155spp	0.041 736spp	0.037 717spp	**0.030** 756spp	**0.017** 756spp	0.023 756spp
Bedroom	0.064 1,727spp	0.035 2,187spp	0.031 719spp	0.031 636spp	**0.026** 705spp	**0.020** 705spp	0.035 705spp
Bookshelf	0.658 2,113spp	0.045 2,749spp	0.054 1,012spp	0.047 1,036spp	**0.030** 1,017spp	**0.025** 1,017spp	0.080 1,017spp
Bottle	0.848 2,110spp	0.088 3,612spp	0.084 1,322spp	0.062 1,451spp	**0.056** 1,280spp	**0.046** 1,280spp	0.163 1,280spp
Cornell Box	0.035 8,614spp	0.009 3,873spp	0.009 906spp	0.009 908spp	**0.005** 1,290spp	**0.006** 1,290spp	0.020 1,290spp
Crytek Sponza	1.340 1,518spp	0.056 2,417spp	0.065 605spp	0.060 533spp	**0.051** 711spp	**0.048** 711spp	0.234 711spp
Glossy Kitchen	1.450 2,092spp	0.071 2,391spp	0.075 988spp	0.049 978spp	**0.045** 875spp	**0.031** 875spp	0.114 875spp
Country Kitchen	0.696 2,070spp	0.068 3,013spp	0.073 1,039spp	0.066 1,026spp	**0.059** 1,014spp	**0.046** 1,014spp	0.067 1,014spp
Necklace	0.289 10,280spp	0.057 9,449spp	0.038 2,958spp	0.033 2,541spp	**0.031** 2,726spp	**0.030** 2,726spp	0.159 2,726spp
Swimming Pool	0.451 4,271spp	0.035 5,771spp	0.040 1,973spp	0.034 1,956spp	**0.031** 1,971spp	**0.028** 1,971spp	0.076 1,971spp
Spaceship	0.017 6,489spp	0.009 7,529spp	0.009 2,775spp	0.008 2,707spp	**0.006** 2,585spp	**0.007** 2,585spp	0.027 2,585spp
Spectral Box	0.030 9,563spp	0.014 4,348spp	0.017 958spp	0.015 946spp	**0.009** 1,373spp	**0.008** 1,373spp	0.019 1,373spp
Sponza Atrium	1.614 1,904spp	0.060 3,007spp	0.049 599spp	**0.040** 550spp	0.042 738spp	**0.022** 738spp	0.066 738spp
Staircase	0.137 1,458spp	0.029 2,553spp	0.023 1,288spp	0.022 1,156spp	**0.019** 943spp	**0.015** 943spp	0.036 943spp
Torus	0.214 12,108spp	0.021 9,470spp	0.020 3,714spp	0.018 2,955spp	**0.015** 3,497spp	**0.014** 3,497spp	0.014 3,497spp
Veach Door	0.532 3,749spp	0.084 2,773spp	0.065 648spp	0.060 603spp	**0.036** 830spp	**0.024** 830spp	0.034 830spp
Veach Lamp	0.532 4,079spp	0.069 2,204spp	0.077 582spp	0.068 537spp	**0.039** 723spp	0.026 723spp	**0.022** 723spp
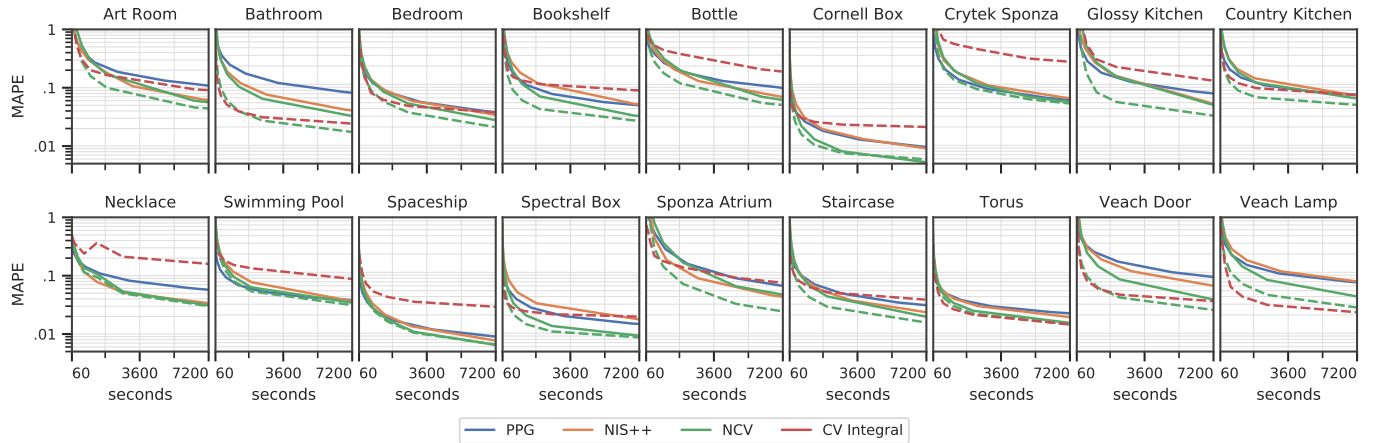
Fig. 9. MAPE convergence plots of practical path guiding (PPG) [Müller et al. 2017], our improved variant (NIS++) of neural importance sampling [Müller et al. 2019], and our neural control variates (NCV). The dashed red line corresponds to using the uncorrected CV integral at the first non-specular vertex. The dashed green line corresponds to the biased variant of our algorithm, where the uncorrected CV integral is used at a vertex determined by the path termination heuristic. It consistently outperforms all other techniques, except for the Torus and the Veach Lamp, where using the CV integral at the first non-specular vertex performs best.

Finally, we demonstrate the *spatial* adaptivity of our model in Figure 11, where we show the learned CV and PDF at several locations in space. Combined with the observations from Figure 10, the spatial adaptivity confirms that the learned CV and PDF capture the full 7-dimensional integrand of the rendering equation with a reasonable accuracy. We visualize the spatio-directional variations of the learned distributions in the supplementary video.

Convergence plots. In Figure 9, we plot MAPE vs. time (seconds) for PPG, NIS++, and our unbiased and biased NCV applications. The unbiased NCVs (green line) are mostly on-par or slightly better than

CV Integral at First Non-Specular Vertex

Fig. 10. Visualization of the learned scattered light field (the CV integral G) from novel viewpoints in several scenes. The light field was trained while rendering the corresponding scenes in Figure 8. As the shown camera views were not used for rendering, they were learned by our neural networks from secondary path vertices. We synthesize the visualizations by evaluating the learned light field for each pixel at the first non-specular path vertex (left). We also show the learned CV (top-right) and sampling PDF (bottom-right) for the given viewing direction at the marked locations (red).

NIS++, except for the BOOKSHELF, CORNELL BOX, SPECTRAL BOX, VEACH DOOR, and VEACH LAMP scenes, where the difference is more pronounced. Adding the heuristic path termination (dashed green line) significantly improves results in most scenes. Interestingly, the almost noise-free CV integral, when applied at the first non-specular path vertex, initially performs much better than the other techniques in terms of MAPE. However, as soon as a sufficient number of samples are drawn, the (nearly) unbiased techniques overtake the significantly biased learned CV integral in most scenes.

8 DISCUSSION AND FUTURE WORK

The use of neural networks makes our technique more computationally expensive than many non-neural approaches such as PPG (see Table 2) or Gaussian [Vorba et al. 2014] and von Mises-Fischer mixture models [Herholz et al. 2019]. It is worth emphasizing that the different methods in the equal-time comparisons do not all use the same hardware: PPG utilizes only the CPU while NIS and NCV

additionally leverage two GPUs and require roughly 3× more power. Nonetheless, even in a light-weight path tracer such as Mitsuba, the per-sample variance reduction of NCV sometimes outweighs the added computational cost and higher power consumption. In cases with high tracing and shading costs—such as in production rendering—the relative overhead of the networks will be smaller and a larger efficiency attainable. The practicality of NCV thus strongly depends on the use case and availability of hardware acceleration.

When the small amount of bias from our path termination heuristic is acceptable, the average path length is reduced by roughly 3× (see Figure 12). Efficiency is therefore improved twofold: lower sample variance *and* much cheaper paths. Developing neural estimators that simulate all light bounces with short paths only, e.g. via Q-learning [Dahm and Keller 2018], may enable synthesizing high-quality images at interactive and real-time rates in the future. Leveraging samples across time may also become an enabler.

Separate learning of α and G. Recall that the control variate is defined as the product $\hat{g} = \alpha \cdot G \cdot \bar{g}$. Since *both* α and G are scaling factors of the control variate, it is possible to combine and learn them as a single variance-minimizing factor $\hat{G}$. The reason for keeping them separate, in our case, is that a variance-minimizing $\hat{G}$ may not necessarily approximate F well. However, we need a good approximation $G \approx F$ for two reasons: (i) to normalize the relative loss terms by G^2 and (ii) to approximate the path-tail contribution by G when heuristically terminating the paths.

Benefit of unbiased control variates. Our unbiased control variates improve the efficiency over pure NIS by 6–17× on a toy problem (Figure 4) whereas they yield a much smaller benefit in light-transport simulations (Table 2 and Figure 13). We suspect that this discrepancy arises from the respective difficulties of the integration problems. In the 2D toy problems, our spectral control variate comes close to a zero-variance configuration, yielding much greater efficiency than pure NIS, which is limited to learning a monochromatic function. In the light-transport simulation, the integrand is higher dimensional (7D) and less well behaved, leading to a larger distance between our models and the zero-variance configuration. This hypothesis is supported by the fact that simpler scenes, such as the SPECTRAL Box, benefit more from our control variates. Importantly, Figure 13 demonstrates that despite the smaller efficiency improvements in the light-transport simulation, unbiased spectral control variates still have a fundamental advantage over monochromatic techniques. Furthermore, the spectrally learned integral of the control variate (see Section 6.1) results in an additional significant efficiency gain.

Approximation power. Figure 14 shows that optimizing the model of the CV integral using 8192spp (all other figures use much fewer spp) enables accurately approximating intricate, high-frequency signals. This suggests that our method is not limited by the approximation power of the employed neural networks, but by the rate of learning. We confirmed this experimentally by increasing the number of sub-flows and by making the neural network bigger. Both resulted in only minor variance reduction—much less than the added computational overhead. For future work, it is therefore of particular interest to investigate means of increasing the training efficiency of the model.
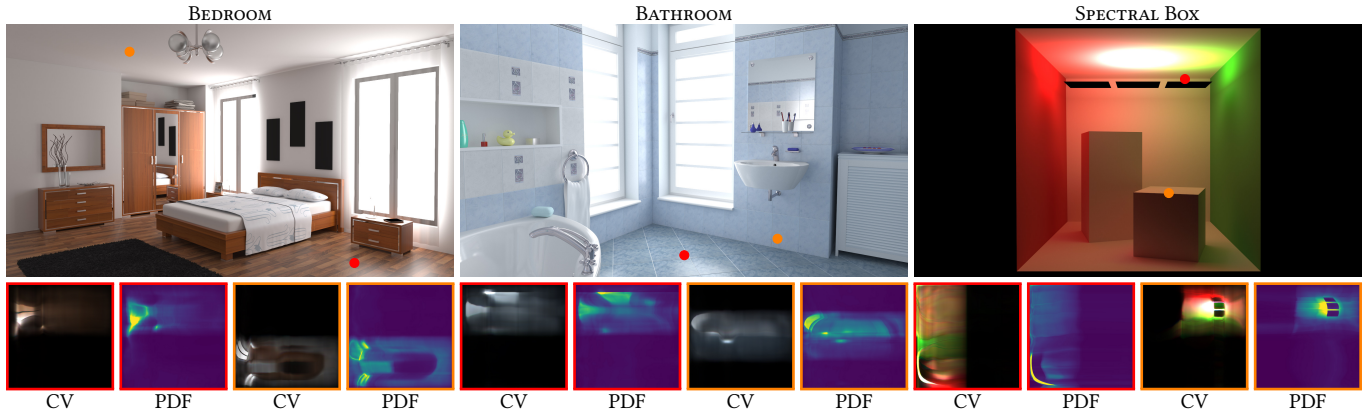
Fig. 11. Visualization of the learned control variates and importance sampling distributions in several scenes. We show CVs (left) and sampling PDFs (right) at two locations (red and orange) per scene. The CVs and PDFs are parametrized in world space via cylindrical coordinates as learned by our neural networks.
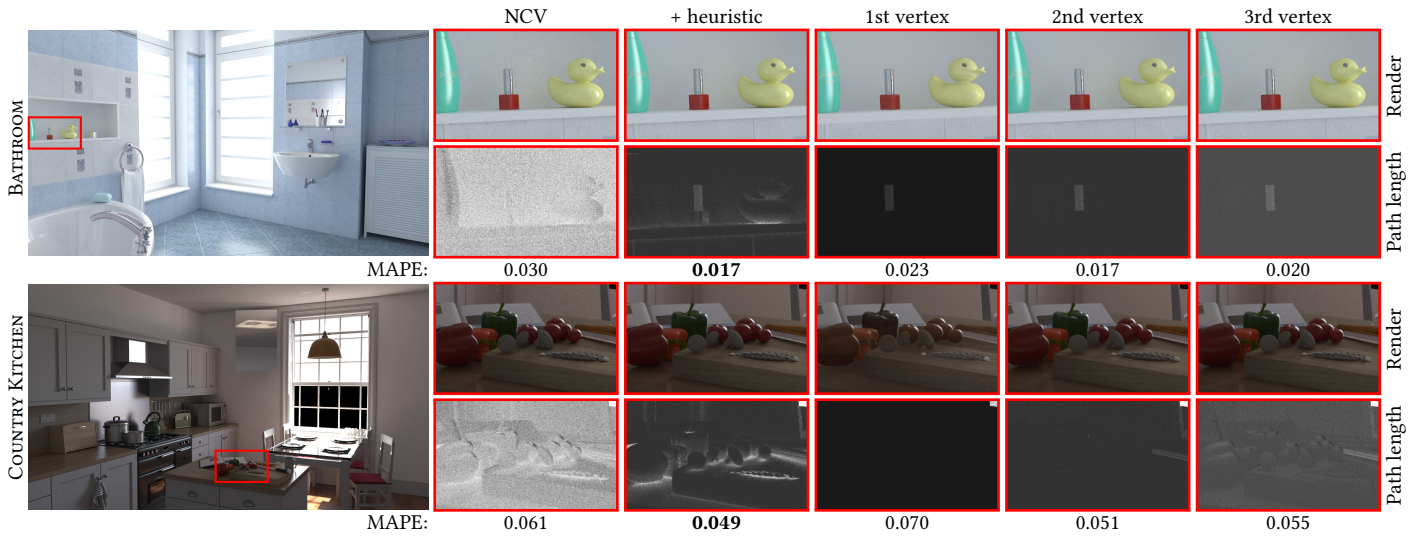

Fig. 12. Different path termination strategies. We compare our unbiased NCVs (no early termination) to our heuristic (Section 6.1) and to terminating at the 1st, 2nd, or 3rd non-specular vertex. The top row shows the resulting images and the bottom row shows the average path length in each pixel (brighter means longer). The heuristic consistently produces less noise than always terminating at the 3rd vertex while simultaneously avoiding visible artifacts in creases and on rough surfaces (as seen when terminating always at the 1st or 2nd vertex).
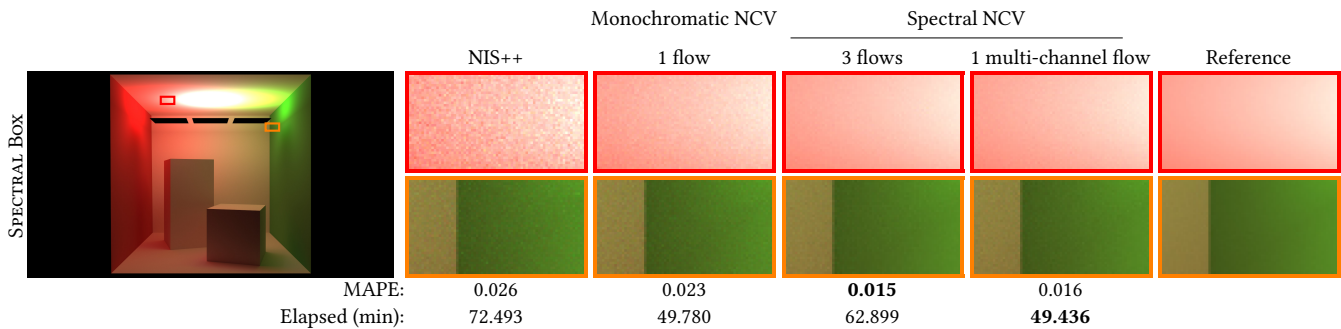
Fig. 13. Comparison of monochromatic and spectral control variates at an equal sampling rate (512 spp). On scenes where the neural control variate is most useful, such as the SPECTRAL BOX shown here, the 3-flows spectral approach can offer a significant improvement over the monochromatic one, but at a higher computational cost. Our multi-channel flow offers a good trade-off between the two approaches. The higher rendering time of NIS is due to the fact that it tends to yield longer paths on the average than the CV approaches in this scene. The impact could be smaller or even reversed in other scenes.
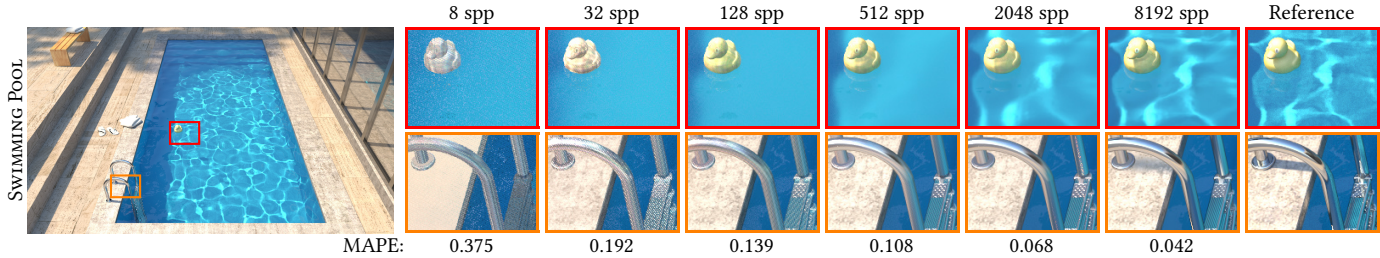
Fig. 14. CV integral training convergence. Training a dedicated CV integral network for longer than 2 hours enables more accurate approximations. This suggests that our method is not limited by the approximation power of the neural networks, but by the speed of their training.

Animated sequences. Figure 10 demonstrates that the neural networks learn a reasonable approximation of the full light field—not just the slice visible by the camera. When rendering an animation, it would thus be reasonable to bootstrap the training of the next frame by initializing the networks using the weights resulting from rendering the previous frame (as opposed to randomly).

8.1 Extensions

Handling of signed integrands. In Section 3, we point out that our parametric neural control variates are non-negative by construction. Inspired by Owen and Zhou's [2000] positivisation trick for importance sampling, we show that one can construct an arbitrary *signed* control variate from two such non-negative control variates.

Let $f(x)$ be a signed integrand. Then the first non-negative control variate $g^+(x)$ shall approximate the positive portion of $f(x)$ and the second non-negative control variate $g^-(x)$ shall approximate the negative portion of $f(x)$. Formally:

$$g^+(x) \approx \max(f(x), 0) \tag{37}$$
$$g^-(x) \approx -\min(f(x), 0) . \tag{38}$$

The signed control variate $g(x) := g^+(x) - g^-(x)$ is defined as the difference of the two control variates. It follows that $g(x)$ approximates $f(x)$ as desired:

$$g(x) \equiv g^+(x) - g^-(x) \approx \max(f(x), 0) + \min(f(x), 0) = f(x) . \tag{39}$$

Adjoint Russian roulette and splitting. Many path tracers utilize Russian roulette to terminate paths and thereby probabilistically avoid evaluating low-contribution samples. Since the neural light field approximation at the primary vertex already contains the majority of the contribution to the pixel value with only little error to be corrected by the remaining path tail, Russian roulette should be aggressively truncating the paths, leading to much greater efficiency. Unfortunately, in our experiments, Russian roulette instead *worsened* the efficiency of our approach. We suspect that this is because it does not take the *variance* of the error correction into account, which is a crucial component of efficiency-optimized Russian roulette [Veach 1997]. Orthogonally, it is possible to improve the accuracy of Russian roulette by taking the adjoint into account [Vorba and Křivánek 2016], however it is difficult to obtain a good estimate of the adjoint. This is because our renderer is not estimating radiance, but the difference to our CV. The appropriate adjoint in this case is the difference integral, which is not readily available.

We leave the investigation of efficiency-optimized Russian roulette within our parametric control variates as future work.

Generalization to volume rendering. Even though we only demonstrated results for surface rendering, we believe our algorithm generalizes to volumetric rendering. First, for estimating in-scattered radiance at a point, the only part of our algorithm that needs to change is the network input listed in Table 1. It is straightforward to add relevant properties of volumetric points to the network input. Then, for estimating radiance along a line through a volume, one may either use an additional 1-D instance of our model, or one may use a form of guided distance sampling [Herholz et al. 2019], complemented by the neural light field G to approximate the true in-scattered radiance.

9 CONCLUSION

We present neural control variates, a model for reducing variance in parametric Monte Carlo integration. The main challenge that we tackle is designing a model with sufficient approximation power that is efficient to evaluate. We achieve this by employing normalizing flows to model the shape of the control variate and a second neural network to infer its integral. To this end, prior works on normalizing flows are extended by developing the multi-channel normalizing flows, which improve the performance of multi-channel integration such as spectral rendering. To further reduce the integration error, we utilize neural importance sampling for estimating the correction term. We describe recipes for jointly optimizing the NCVs and the residual NIS using (i) a theoretically optimal variance loss, and (ii) an empirical composite loss for robust optimization.

We analyzed the performance of neural control variates in the setting of photorealistic image synthesis. The NCVs yield notable improvements and perform better, on the average, than state-of-the-art competitors in both equal-time and equal-sample-count settings. While our unbiased application of NCVs only provides a small efficiency boost, it enables a biased algorithm that improves efficiency significantly. We expect our performance to grow further as the considerable cost of neural networks decreases with the advance of models and computer hardware. While we demonstrate the utility of our approach for path tracing, we think it will extend well to many rendering algorithms that employ Monte Carlo integration, and we expect it to be applicable beyond light-transport algorithms due to its fundamental nature.

Convergence of data-driven and physically based simulation. Our work connects neural approximation and unbiased simulation. While

data-driven approaches can obtain realistic results, correcting their errors is tedious. Physically based integrators, on the other hand, provide accurate solutions, albeit at excessive cost. We show that the mechanism of control variates allows for combining a data-driven, high-quality neural approximation with an accurate, physically-based integrator, which can be used on demand to merely correct the errors. The challenge for such future developments will be defining G and g such that $G = \int g(x)\,dx$ is preserved while both are efficient to evaluate.

ACKNOWLEDGMENTS

A VARIANCE OF THE CONTROL VARIATE ESTIMATOR

In order to derive the variance of the control variate estimator

$$\langle F \rangle = G(\theta_{\hat{G}}) + \frac{f(X, Z)}{p(X, Z; \theta_p)} - \frac{\hat{g}(X; \theta_{\hat{g}})}{p(X; \theta_p)}, \tag{40}$$

we recall that $\mathbb{V}[X + c] = \mathbb{V}[X]$ for any constant c. Hence, subtracting the constant $\hat{G}(\theta_{\hat{G}})$ on both sides and proceeding with the definition of variance, we have:

$$\mathbb{V}[\langle F \rangle] = \mathbb{V}[\langle F \rangle - \hat{G}(\theta_{\hat{G}})]$$
$$= \underbrace{\mathbb{E}\left[\left(\langle F \rangle - \hat{G}(\theta_{\hat{G}})\right)^2\right]}_{=: U} - \underbrace{\mathbb{E}\left[\langle F \rangle - \hat{G}(\theta_{\hat{G}})\right]^2}_{=: V}. \tag{41}$$

Then the expectation of the square results in the following double integral:

$$\mathbb{E}\left[\left(\langle F \rangle - \hat{G}(\theta_{\hat{G}})\right)^2\right]$$
$$= \mathbb{E}\left[\left(\frac{f(X, Z)}{p(X, Z; \theta_p)} - \frac{\hat{g}(X; \theta_{\hat{g}})}{p(X; \theta_p)}\right)^2\right]$$
$$= \int_{\mathcal{D}} \int_{\mathcal{P}} \left(\frac{f(x, z)}{p(x, z; \theta_p)} - \frac{\hat{g}(x; \theta_{\hat{g}})}{p(x; \theta_p)}\right)^2 p(x, z; \theta_p)\,dz\,dx$$
$$= \int_{\mathcal{D}} \int_{\mathcal{P}} \left(\frac{f(x, z)}{p(x; \theta_p)p(z|x)} - \frac{\hat{g}(x; \theta_{\hat{g}})}{p(x; \theta_p)}\right)^2 p(x; \theta_p)p(z|x)\,dz\,dx$$
$$= \int_{\mathcal{D}} \int_{\mathcal{P}} \left(\frac{f(x, z)}{p(z|x)} - \hat{g}(x; \theta_{\hat{g}})\right)^2 \frac{p(z|x)}{p(x; \theta_p)}\,dz\,dx = U. \tag{42}$$

The squared expectation (second term in Equation (41)) simplifies to

$$\left(\mathbb{E}[\langle F \rangle] - \hat{G}(\theta_{\hat{G}})\right)^2 = \left(F - \hat{G}(\theta_{\hat{G}})\right)^2 = V. \tag{43}$$

Putting the U and V terms together yields Equation (17).

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/

Roland Assaraf and Michel Caffarel. 1999. Zero-Variance Principle for Monte Carlo Algorithms. *Phys. Rev. Lett.* 83 (Dec 1999), 4682–4685. Issue 23. https://doi.org/10.1103/PhysRevLett.83.4682

Andrea Barth, Christoph Schwab, and Nathaniel Zollinger. 2011. Multi-level Monte Carlo Finite Element method for elliptic PDEs with Stochastic Coefficients. *Numer. Math.* 119, 1 (2011), 123–161. https://doi.org/10.1007/s00211-011-0377-0

Philippe Bekaert, Philipp Slusallek, Ronald Cools, Vlastimil Havran, and Hans-Peter Seidel. 2003. A custom designed Density Estimation Method for Light Transport. *MPI-I-2003-4-004* (April 2003).

Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 2013. 5D Covariance Tracing for Efficient Defocus and Motion Blur. *ACM Trans. Graph.* 32, 3, Article Article 31 (July 2013), 18 pages. https://doi.org/10.1145/2487228.2487239

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Mark Broadie and Paul Glasserman. 1998. *Risk Management and Analysis, Volume 1: Measuring and Modelling Financial Risk.* Wiley, New York, Chapter Simulation for option pricing and risk management, 173–208.

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

Petrik Clarberg and Tomas Akenine-Möller. 2008. Exploiting Visibility Correlation in Direct Illumination. *Computer Graphics Forum* 27, 4 (2008), 1125–1136. https://doi.org/10.1111/j.1467-8659.2008.01250.x

Ken Dahm and Alexander Keller. 2018. Learning Light Transport the Reinforced Way. In *Monte Carlo and Quasi-Monte Carlo Methods*, Art B. Owen and Peter W. Glynn (Eds.). Springer International Publishing, 181–195.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516* (Oct. 2014).

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density Estimation using Real NVP. *arXiv:1605.08803* (March 2016).

Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-Chi Lai. 2006. Optimizing Control Variate Estimators for Rendering. *Computer Graphics Forum* 25, 3 (2006), 351–358.

Iliyan Georgiev, Zackary Misso, Toshiya Hachisuka, Derek Nowrouzezahrai, Jaroslav Křivánek, and Wojciech Jarosz. 2019. Integral formulations of volumetric transmittance. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38, 6 (Nov. 2019). https://doi.org/10/dffn

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. MADE: Masked Autoencoder for Distribution Estimation. In *International Conference on Machine Learning*. 881–889.

Michael B. Giles. 2008. Improved Multilevel Monte Carlo Convergence using the Milstein Scheme. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Alexander Keller, Stefan Heinrich, and Harald Niederreiter (Eds.). Springer, Berlin, Heidelberg, 343–358. https://doi.org/10.1007/978-3-540-74496-2_20

Michael B. Giles. 2013. Multilevel Monte Carlo Methods. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, Josef Dick, Y. Frances Kuo, W. Gareth Peters, and H. Ian Sloan (Eds.). Springer, Berlin, Heidelberg, 83–103. https://doi.org/10.1007/978-3-642-41095-6_4

Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proc. 13th International Conference on Artificial Intelligence and Statistics* (May 13–15). JMLR.org, 249–256.

Peter W. Glynn and Roberto Szechtman. 2002. Some New Perspectives on the Method of Control Variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, Kai-Tai Fang, Harald Niederreiter, and Fred J. Hickernell (Eds.). Springer, Berlin, Heidelberg, 27–49. https://doi.org/10.1007/978-3-642-56046-0_3

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. 2018. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. *International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stefan Heinrich. 1998. Monte Carlo Complexity of Global Solution of Integral Equations. *Journal of Complexity* 14, 2 (1998), 151 – 175. https://doi.org/10.1006/jcom.1998.0471

Stefan Heinrich. 2000. The Multilevel Method of Dependent Tests. In *Advances in Stochastic Simulation Methods*. Birkhäuser Boston, Boston, MA, 47–61. https://doi.org/10.1007/978-1-4612-1318-5_4

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P. A. Lensch, and Jaroslav Křivánek. 2019. Volume Path Guiding Based on Zero-Variance Random Walk Theory. *ACM Trans. Graph.* 38, 3, Article 25 (June 2019), 19 pages. https://doi.org/10.1145/3230635

Pedro Hermosilla, Sebastian Maisch, Tobias Ritschel, and Timo Ropinski. 2019. Deep-learning the Latent Space of Light Transport. *Computer Graphics Forum* 38, 4 (2019).

Timothy C. Hesterberg and Barry L. Nelson. 1998. Control Variates for Probability and Quantile Estimation. *Management Science* 44, 9 (Sept. 1998), 1295–1312. https:

//doi.org/10.1287/mnsc.44.9.1295

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

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167* (2015).

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

Simon Kallweit, Thomas Müller, Brian McWilliams, Markus Gross, and Jan Novák. 2017. Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Trans. Graph.* 36, 6, Article 231 (Nov. 2017), 11 pages. https://doi.org/10.1145/3130800.3130880

Alexander Keller. 2001. Hierarchical Monte Carlo Image Synthesis. *Mathematics and Computers in Simulation* 55, 1–3 (2001), 79 – 92. https://doi.org/10.1016/S0378-4754(00)00248-2 The Second {IMACS} Seminar on Monte Carlo Methods.

Angelien Kemna and Ton Vorst. 1990. A Pricing Method for Options based on Average Asset Values. *Journal of Banking & Finance* 14, 1 (1990), 113–129. https://doi.org/10.1016/0378-4266(90)90039-5

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (June 2014).

Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv:1807.03039* (July 2018).

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved Variational Inference with inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems*. 4743–4751.

Ivan Kobyzev, Simon Prince, and Marcus A. Brubaker. 2019. Normalizing Flows: An Introduction and Review of Current Methods. arXiv:stat.ML/1908.09257

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. 1994. The Ambient Term as a Variance Reducing Technique for Monte Carlo Ray Tracing. In *Proc. EGWR*. 163–171.

Eric P. Lafortune and Yves D. Willems. 1995. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Proc. EGWR*. 11–20.

Stephen S. Lavenberg, Thomas L. Moeller, and Peter D. Welch. 1982. Statistical Results on Control Variables with Application to Queueing Network Simulation. *Operations Research* 30, 1 (1982), 182–202. https://doi.org/10.1287/opre.30.1.182

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. arXiv:cs.CV/1803.04189

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323020

Maxim Maximov, Laura Leal-Taixe, Mario Fritz, and Tobias Ritschel. 2019. Deep Appearance Maps. In *The IEEE International Conference on Computer Vision (ICCV)*.

Abhimitra Meka, Christian Häne, Rohit Pandey, Michael Zollhöfer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. 2019. Deep Reflectance Fields: High-quality Facial Reflectance Field Inference from Color Gradient Illumination. *ACM Trans. Graph.* 38, 4, Article 77 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323027

Antonietta Mira, Reza Solgi, and Daniele Imparato. 2013. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing* 23 (2013), 653–662.

Thomas Müller. 2019. "Practical Path Guiding" in Production. In *ACM SIGGRAPH Courses: Path Guiding in Production, Chapter 10*. ACM, New York, NY, USA, 18:1–18:77. https://doi.org/10.1145/3305366.3328091

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

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM Trans. Graph.* 38, 5, Article 145 (Oct. 2019), 19 pages. https://doi.org/10.1145/3341156

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen-Space Shading. 36, 4 (2017).

Barry L. Nelson. 1990. Control Variate Remedies. *Operations Research* 38, 6 (1990), 974–992. https://doi.org/10.1287/opre.38.6.974

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual Ratio Tracking for Estimating Attenuation in Participating Media. *ACM Trans. Graph.* 33, 6 (Nov. 2014). https://doi.org/10.1145/2661229.2661292

C. Oates, M. Girolami, and N. Chopin. 2014. Control functionals for Monte Carlo integration. *Journal of The Royal Statistical Society Series B-statistical Methodology* 79 (2014), 695–718.

Art Owen and Yi Zhou. 2000. Safe and Effective Importance Sampling. *J. Amer. Statist. Assoc.* 95, 449 (2000), 135–143. http://www.jstor.org/stable/2669533

George Papamakarios, Iain Murray, and Theo Pavlakou. 2017. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*. 2338–2347.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2019. Normalizing Flows for Probabilistic Modeling and Inference. arXiv:stat.ML/1912.02762

Vincent Pegoraro, Carson Brownlee, Peter S. Shirley, and Steven G. Parker. 2008a. Towards Interactive Global Illumination Effects via Sequential Monte Carlo Adaptation. In *Proceedings of the 3rd IEEE Symposium on Interactive Ray Tracing*. 107–114.

Vincent Pegoraro, Ingo Wald, and Steven G. Parker. 2008b. Sequential Monte Carlo Adaptation in Low-Anisotropy Participating Media. *Computer Graphics Forum* 27, 4 (2008), 1097–1104.

Matt Pharr, Wenzel Jacob, and Greg Humphreys. 2016. *Physically Based Rendering - From Theory to Implementation*. Morgan Kaufmann, Third Edition.

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global Illumination with Radiance Regression Functions. *ACM Trans. Graph.* 32, 4, Article 130 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2462009

Danilo Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*. 1530–1538.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space Control Variates for Rendering. *ACM Trans. Graph.* 35, 6, Article 169 (Nov. 2016), 12 pages. https://doi.org/10.1145/2980179.2982443

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2011. Adaptive Sampling and Reconstruction Using Greedy Error Minimization. *ACM Trans. Graph.* 30, 6, Article 159 (Dec. 2011), 12 pages. https://doi.org/10.1145/2070781.2024193

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. 2018. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *CVPR*.

Charles Stein. 1972. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. University of California Press, Berkeley, Calif., 583–602. https://projecteuclid.org/euclid.bsmsp/1200514239

László Szécsi, Mateu Sbert, and László Szirmay-Kalos. 2004. Combined Correlated and Importance Sampling in Direct Light Source Computation and Environment Mapping. *Computer Graphics Forum* 23 (2004), 585–594.

László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. 2011. Free Path Sampling in High Resolution Inhomogeneous Participating Media. *Computer Graphics Forum* 30, 1 (2011), 85–97.

Esteban Tabak and Cristina V. Turner. 2013. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics* 66, 2 (2013), 145–164. https://doi.org/10.1002/cpa.21423

Esteban Tabak and Eric Vanden Eijnden. 2010. Density Estimation by dual Ascent of the Log-Likelihood. *Communications in Mathematical Sciences* 8, 1 (2010), 217–233.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhöfer. 2020. State of the Art on Neural Rendering. arXiv:cs.CV/2004.03805

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019. Deferred Neural Rendering: Image Synthesis Using Neural Textures. *ACM Trans. Graph.* 38, 4, Article 66 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323035

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

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proc. SIGGRAPH*. 419–428. https://doi.org/10.1145/218380.218498

Delio Vicini, Vladlen Koltun, and Wenzel Jakob. 2019. A Learned Shape-Adaptive Subsurface Scattering Model. *ACM Trans. Graph.* 38, 4, Article 127 (July 2019), 15 pages. https://doi.org/10.1145/3306346.3322974

Marc Sabate Vidales, David Siska, and Lukasz Szpruch. 2018. Unbiased deep solvers for parametric PDEs. *arXiv:1810.05094* (Oct. 2018).

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph.* 33, 4 (Aug. 2014).

Jiří Vorba and Jaroslav Křivánek. 2016. Adjoint-Driven Russian Roulette and Splitting in Light Transport Simulation. *ACM Trans. Graph.* 35, 4 (jul 2016).

Ruosi Wan, Mingjun Zhong, Haoyi Xiong, and Zhanxing Zhu. 2019. Neural Control Variates for Variance Reduction. *arXiv:1806.00159* (Oct. 2019).

Tomoya Yamaguchi, Tatsuya Yatagawa, and Shigeo Morishima. 2018. Efficient Metropolis Path Sampling for Material Editing and Re-rendering. In *Pacific Graphics Short Papers*, Hongbo Fu, Abhijeet Ghosh, and Johannes Kopf (Eds.). The Eurographics Association. https://doi.org/10.2312/pg.20181271

Quan Zheng and Matthias Zwicker. 2019. Learning to Importance Sample in Primary Sample Space. *Computer Graphics Forum* 38, 2 (2019), 169–179. https://doi.org/10.1111/cgf.13628