

Neural Control Variates

THOMAS MÜLLER, NVIDIA
FABRICE ROUSSELLE, NVIDIA
JAN NOVÁK, NVIDIA
ALEXANDER KELLER, NVIDIA

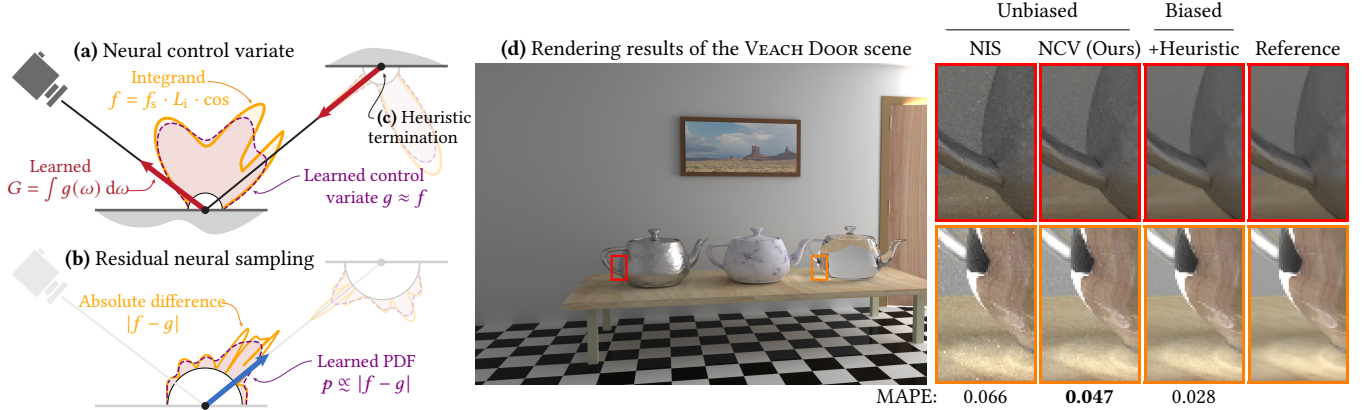


Fig. 1. When applied to light transport simulation, our neural control variate algorithm (a) learns an approximation G of the scattered light field and corrects the approximation error by estimating the difference between the original integrand f and the corresponding learned control variate g . This 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 PDF p (b). We also provide a heuristic (c) to terminate paths early by using the approximation G . 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 neural importance sampling (NIS) and our neural control variates (NCV) with and without our path-termination heuristic. At equal time (2h), our unbiased technique has 29% lower mean absolute percentage error (MAPE), whereas our biased algorithm reduces MAPE by an additional 40%. This error reduction roughly amounts to 10× greater efficiency.

We propose the concept of neural control variates (NCV) for unbiased variance reduction in parametric Monte Carlo integration for solving integral equations. 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 insignificant visible bias.

Authors' addresses: Thomas Müller, NVIDIA, tmuller@nvidia.com; Fabrice Rousselle, NVIDIA, frousselle@nvidia.com; Jan Novák, NVIDIA, jnovak@nvidia.com; Alexander Keller, NVIDIA, akeller@nvidia.com.

© 2020 Copyright held by the owner/author(s). This is the author's version of the work. It is posted here for your personal use. Not for redistribution.

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Ray tracing**; *Supervised learning by regression*; *Reinforcement learning*; • **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:

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

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. $G = \int_{\mathcal{D}} g(x) dx$. 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 key challenge becomes representing it in a form that permits (efficiently) computing its integral, $G = \int_{\mathcal{D}} g(x) dx$. We make the key observation that this difficulty is completely sidestepped if we decompose the control variate into its normalized form—the shape $\bar{g}(x)$ —and the 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, in contrast to standard neural networks, 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 $\int_{\mathcal{D}} f(x) - \alpha \cdot g(x) dx$ typically carries less energy than the original integral $\int_{\mathcal{D}} f(x) dx$. However, the residual *integrand* may feature shapes that are hard to sample with hand-crafted distributions; this is why many prior works struggled in combining control variates with importance sampling in graphics.

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

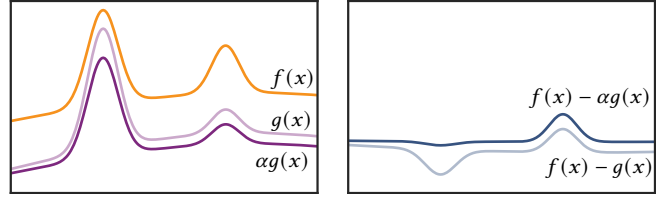


Fig. 2. Computing an integral F of a function $f(x)$ with the help of a control variate $g(x)$ (left) amounts to using a known (or efficient-to-compute) integral $G = \int_{\mathcal{D}} g(x) dx$ and adding the integrated difference $f(x) - g(x)$ (right). The overall variance may be reduced by minimizing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

computational overhead of using potentially *many* control variates, we develop a novel type of 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 $\text{Cov}(f, g) / \text{Var}(g)$ (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 due to the limited approximation power of their data structure. Others proposed to apply CVs to carefully chosen subproblems, such as estimating direct illumination [Clarberg and Akenine-Möller 2008; Fan et al. 2006; Szécsi et al. 2004], sampling free-flight distances in participating media [Novák et al. 2014; Szirmay-Kalos et al. 2011], or unbiased denoising and re-rendering [Rousselle et al. 2016].

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 fact, they are capable of universal function approximation [Hornik et al. 1989]. 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 mip-level 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. While unbiased results may not have been the priority in the specific applications targeted previously, we view our neural control variates as a step towards bringing data-driven and physically-based image synthesis 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 the score function. Alternatively to our usage of normalizing flows, [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. $\mathbb{E}_{x \sim p}[s(x)] = 0$, trivially allowing its use as a control variate of a stochastic estimator. Wan et al. [2019] suggest a transformation of the score function that is parameterized by neural networks such that the zero mean is preserved, resulting in a neural control variate. In contrast to our use of normalizing flows, using the score function 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.

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

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

depending on the parameter 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 5, $F(y)$ is the reflected radiance field 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 7.

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; $\theta_g := \theta_{\bar{g}} \cup \theta_G$. 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. Section 4 then elaborates on recipes for optimizing their 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 $\bar{g}(x, y; \theta_{\bar{g}})$ 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 computational graph that represents a differentiable, multi-dimensional, compound mapping for transforming probability densities. The mapping comprises L bijective warping functions $\bar{h} = h_L \circ \dots \circ h_2 \circ h_1$; it is therefore also bijective as a whole. The warping functions

$h: \mathcal{X} \rightarrow \mathcal{X}'$ 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 $x_1 = x$. 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 $\hat{x} = \bar{h}(x)$ 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 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.

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 dimensionality as g to be the latent space $\mathcal{L}$. The normalized CV is then modeled as

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

It is worth noting that the product on the right-hand side is normalized by construction: the probability density $p_{\mathcal{L}}$ is normalized

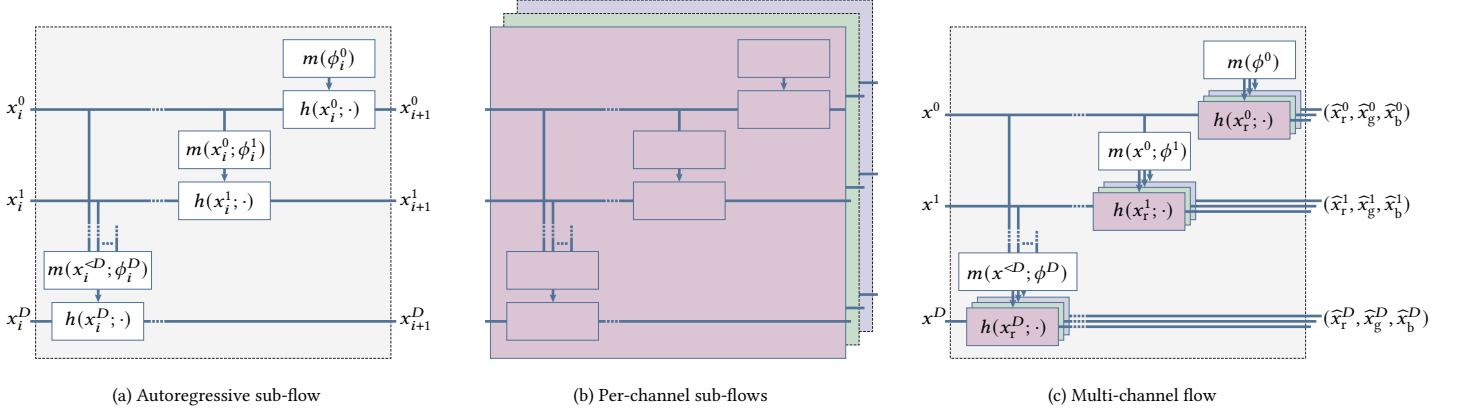


Fig. 3. We model the shape of the control variate using bijective transformations assuming autoregressive structure (a) as proposed by Kingma et al. [2016] in the context of probabilistic generative models. Concatenating multiple autoregressive blocks (sub-flows) increases the expressivity of the model. 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 5, we propose to use a single network across all channels (c) to keep the evaluation cost largely agnostic to the channel count.

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 :

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

where the superscript $< d$ denotes the preceding dimensions and ϕ_i^d are network parameters. 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, 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). 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 flows 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 *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. For stable optimization, we exponentiate the neural network output, allowing it to output a high-dynamic range of values while internally operating on a numerically better behaved low dynamic range.

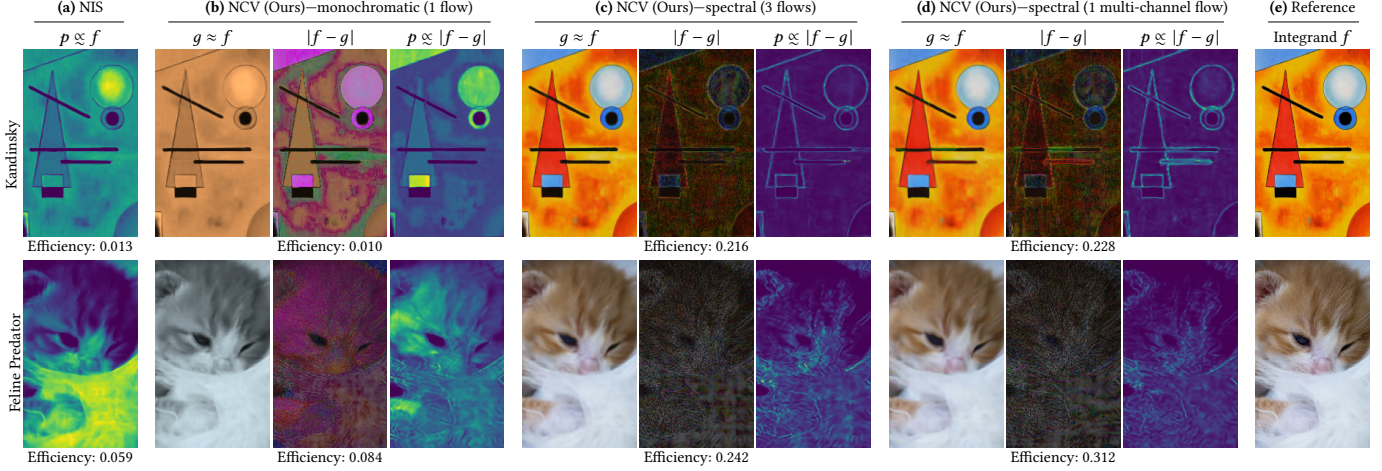


Fig. 4. Comparison of neural importance sampling (a) [Müller et al. 2019] and various flavors of our neural control variates (b, c, d) on two toy integration problems (rows). The integrands are 2D images with 3 color channels (e); the goal is to Monte Carlo estimate the average color of the images, i.e. the integration domain is 2-dimensional. We report the Monte Carlo efficiency, defined as $(\mathbb{V}[\langle F \rangle] \cdot \text{runtime})^{-1}$, of using the different techniques. 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 performs best, because its fit only has slightly worse quality than the three flows while being much cheaper to evaluate and train.

It is worth noting that the combination of the exponentiated network output (always positive) and the normalizing flow for the CV shape constrains the CV to be a non-negative function; negative values are excluded by design. While this is perfect for our light transport application in Section 5, signed integrands can still be handled using an approach described in Section 7.

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 $\text{Cov}(f(x, y), g(x, y)) / \text{Var}(g(x, y))$ [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 $\alpha(y; \theta_\alpha)$, which is trained to output the appropriate contribution of the CV in dependence on the parameter y . In the following section, we will contribute a loss function for optimizing the neural network $\alpha(y; \theta_\alpha)$ from Monte Carlo estimates such that it minimizes variance.

4 MONTE CARLO INTEGRATION WITH NCV

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

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

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:

$$\hat{G}(\theta_{\hat{G}}) := \alpha(\theta_\alpha) \cdot G(\theta_G); \quad \theta_{\hat{G}} := \theta_\alpha \cup \theta_G, \quad (10)$$

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

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

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

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

4.1 Importance Sampling of the Residual

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 $|f(X) - \hat{g}(X; \theta_{\hat{g}})|$, which strongly correlates with the parametric CV $\hat{g}(X; \theta_{\hat{g}})$. Since the CV will be optimized progressively, the correction term will evolve over time. In the ideal case, the absolute difference $|f(x) - g(x)|$ 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) [Veatch and Guibas 1995] with learned probabilities for selecting each of the two 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 outputs) 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 point during the training process. The final PDF reads:

$$p(x; \theta_p) = (1 - c(x; \theta_c)) p_{\mathcal{U}}(x) + c(x; \theta_c) p_{\text{NIS}}(x; \theta_{\text{NIS}}), \quad (13)$$

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

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) our multi-channel flow. Our 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.

4.2 Minimizing the 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 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

$$\begin{aligned} \mathbb{V}[\langle F \rangle] &= \mathbb{E}[\langle F \rangle^2] - \mathbb{E}[\langle F \rangle]^2 \\ &= \int_{\mathcal{D}} \frac{(f(x) - \hat{g}(x; \theta_{\hat{g}}))^2}{p(x; \theta_p)} dx - (F - \hat{G}(\theta_{\hat{G}}))^2 \end{aligned} \quad (14)$$

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

¹For 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.

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 is equal 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 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 5, 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):

$$\begin{aligned} \mathbb{V}[\langle F \rangle] &= \int_{\mathcal{D}} \int_{\mathcal{P}} \left(\frac{f(x, z)}{p(z|x)} - \hat{g}(x; \theta_{\hat{g}}) \right)^2 \frac{p(z|x)}{p(x; \theta_p)} dz dx \\ &\quad - (F - \hat{G}(\theta_{\hat{G}}))^2; \end{aligned} \quad (17)$$

see Appendix A for a complete derivation.

Finding optimal $\theta_{\hat{g}}$, $\theta_{\hat{G}}$, θ_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 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 autodifferentiation 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:

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

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 $\langle \nabla \rangle$, 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 $f(X)$:

$$\langle \nabla[\langle F \rangle] \rangle = \frac{(\langle f(X) \rangle - \alpha(\theta_\alpha) g(X; \theta_g))^2}{p(X; \theta_p) q(X)} - \left(\frac{\langle f(X) \rangle}{q(X)} - \alpha(\theta_\alpha) G(\theta_G) \right)^2, \quad (19)$$

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 autograd. Unfortunately, despite being theoretically optimal, our empirical analysis revealed poor performance when optimizing with this loss.

4.3 Composite Loss for Stable Optimization

The variance of the parametric estimator, Equation (17), can be zero for an entire family of configurations of θ_α , θ_g , θ_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:

$$G(\theta_G) = F, \quad (20)$$

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

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

$$\alpha(\theta_\alpha) = 1. \quad (23)$$

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:

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

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

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

where $\text{sg}(x)$ indicates that x is treated as a constant, i.e. no gradients w.r.t. it are computed. 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

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

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 our central Section 5. The relative $\mathcal{L}^2$ loss achieves by far 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, $\bar{g}$:

$$\mathcal{L}_H(\bar{f}, \bar{g}; \theta_g) = - \int_{\mathcal{D}} \bar{f}(x) \log(\bar{g}(x; \theta_g)) dx. \quad (27)$$

The main caveat of the cross entropy is that it requires normalizing the integrand, which we can not do, because we do not know F . Thankfully, the normalizing constant disappears from the cross entropy when using certain optimizers (e.g. Adam [Kingma and Ba 2014]) as observed by Müller et al. [2019]. With this observation, an MC estimator of the cross entropy that can be fed to automatic differentiation reads

$$\langle \mathcal{L}_H(\bar{f}, \bar{g}; \theta_g) \rangle = - \frac{\langle f(X) \rangle}{q(X)} \log(\bar{g}(X; \theta_g)). \quad (28)$$

Sampling distribution optimization. Our parametric sampling distribution is also based on a normalizing flow and 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 the flow to approximate the normalized absolute difference in Equation (22). Once again, the normalization constant in the cross entropy loss can be dropped. In addition, we approximate the difference $|f(x) - g(x; \theta_g)|$ using the biased estimator

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

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

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

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

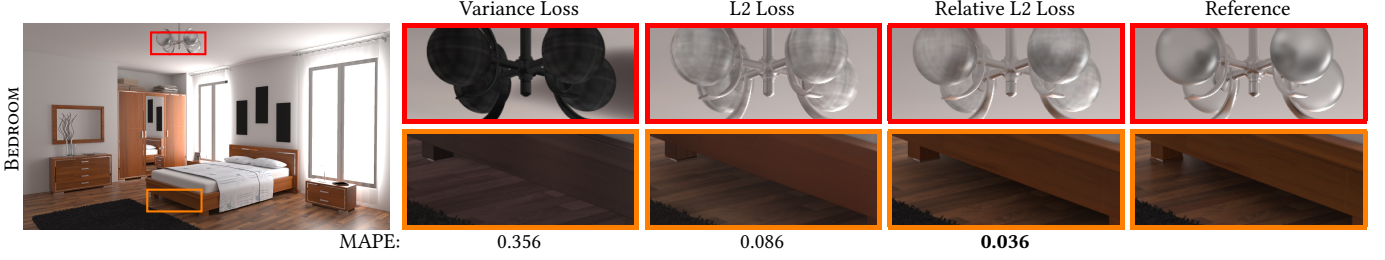


Fig. 5. Control Variate integral optimization. Using the estimator variance loss (left) to optimize the CV integral G effectively merges it with the α coefficient, resulting in a darker output where the CV shape is a poor match for the target. Using the $\mathcal{L}^2$ loss (middle) decouples the CV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

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:

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

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 $\mathcal{L}_V(\theta_\alpha)$ for optimizing θ_α :

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

Since the individual components of the loss in Equation (24) act on disjoint sets of parameters, we use different instances of the Adam optimizer [Kingma and Ba 2014] to allow for adjusting the learning rate per component.

5 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

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

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 will 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 $\hat{g}(\mathbf{x}, \omega, \omega_i; \theta_{\hat{g}})$ approximates the normalized integrand. In analogy to Equation (12), a one-sample MC estimator of Equation (32) with the trainable CV from Section 3 then reads:

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

We made one small modification to $p(\Omega|\mathbf{x}, \omega; \theta_p)$: instead of mixing NIS with uniform sampling as proposed in Section 4.1, 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:

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

Figure 6 visualizes how each component of our trainable CVs fits into the light-transport integral equation.

5.1 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 7 (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

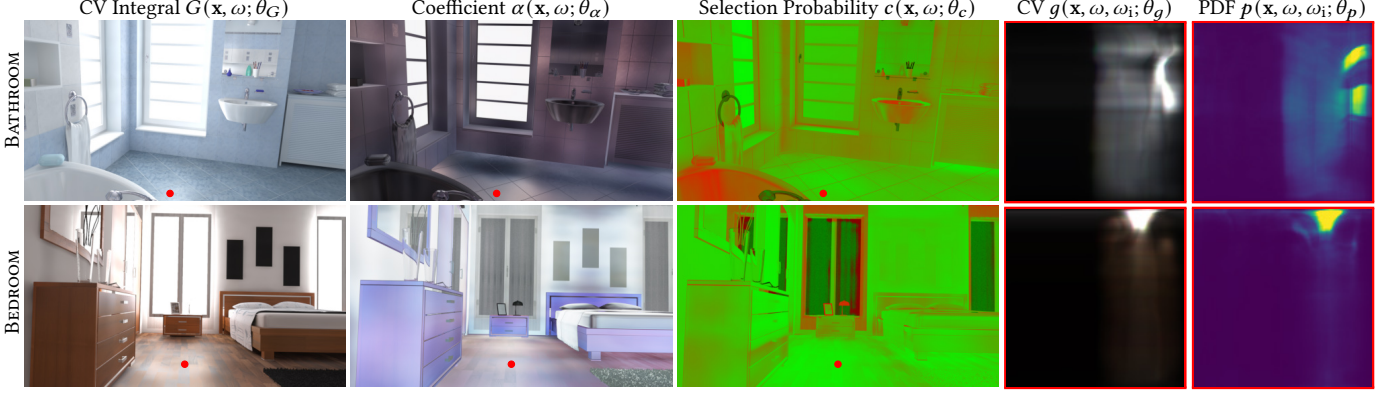


Fig. 6. All learned components of our method when applied to light transport simulation: We visualize the learned CV integral $G(\mathbf{x}, \omega; \theta_G)$, the coefficient $\alpha(\mathbf{x}, \omega; \theta_\alpha)$, and selection probability $c(\mathbf{x}, \omega; \theta_c)$ at the primary vertices (first non-delta interaction) of each pixel. Furthermore, we show the directionally resolved learned CV $g(\mathbf{x}, \omega, \omega_i; \theta_g)$ 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.

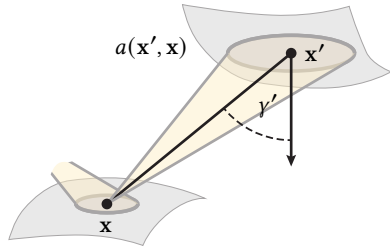
strike a good balance between visual quality and computation cost (“Biased NCV (Ours)” column in Figure 7).

We utilize a simple criterion for ignoring the correction term, i.e. approximating L_s by the neural light field G . 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 using the neural light field G instead.

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

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

around the next path vertex $\mathbf{x}'$, where γ' is the angle of incidence 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 by summing the square root of the area spreads at consecutive vertices and re-squaring:



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

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 6.

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.

This is closely related to the unbiased stochastic termination of paths via Russian roulette, which we discuss in Section 7.

5.2 Implementation

We implemented our neural control variates as well as neural importance sampling within Tensorflow [Abadi et al. 2015]. Our 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 our CPU to perform light-transport computations and two GPUs to perform our 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, 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]: Mitsuba first queries the MIS selection probabilities $c(\mathbf{x}, \omega; \theta_{f_s})$ and

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

$c(\mathbf{x}, \omega; \theta_{\text{NIS}})$. Next, according to the selection probabilities, Mitsuba probabilistically selects either BSDF sampling or NIS. If BSDF sampling is selected, Mitsuba will query the NIS PDF for the sampled direction ω_i , whereas if NIS is selected, Mitsuba will query a sample of ω_i via NIS. Then, we use our neural control variates: Mitsuba queries 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 vertex’s metadata, is put into a ringbuffer that keeps track of the past 1 048 576 vertices. The training GPU continuously samples uniformly random training batches from the ringbuffer to decorrelate paths that arose in close proximity in the image plane.

Specular BSDFs. BSDFs with Dirac-delta components (henceforth referred to as “specular”) typically require special treatment because they are not square integrable and because of the limitations of IEEE floating-point numbers. 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, our selection probability c is used in the regular 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 will have to be treated as zero. Otherwise (i.e. when either NIS or a smooth BSDF component is sampled), one will have to apply our neural control variates, 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 must be 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 ; $i \in \{0, \dots, M\}$, 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 a robust numerical estimate of their mean pixel variance [Müller 2019] in order to limit the impact of the high variance of the 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, denoted by $\text{ob}(x)$.

Network and flow architecture. For all normalizing flows, i.e. the multi-channel flow for the CV shape as well as the regular flow for NIS, we use the piecewise-quadratic warp proposed by [Müller et al. 2019] with 64 bins and a uniform latent distribution $p_{\mathcal{L}}(x') \equiv p_{\mathcal{U}}(x')$. Furthermore, both flows (multi-channel and NIS) 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: they are fully-connected residual networks [He et al. 2016] with 2 residual blocks that each have 2 layers with 256 neurons.

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

We minimize Equation (24) using Adam [Kingma and Ba 2014], a gradient-descent technique that converges to the loss’ minimum even when fed with noisy, unnormalized estimates of the loss gradient, as long as those estimates are unbiased.

We 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 their learned distributions exhibited prolonged fluctuations in their later stages of training. To isolate the benefits of the learning-rate decay, we also list the results of neural importance sampling *without* the learning-rate decay in Table 2.

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

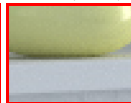
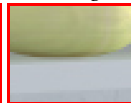
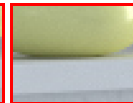
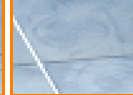
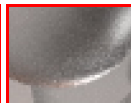
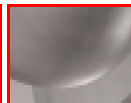
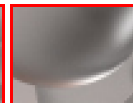
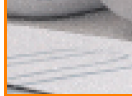
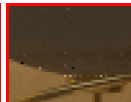
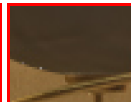
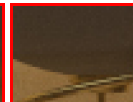
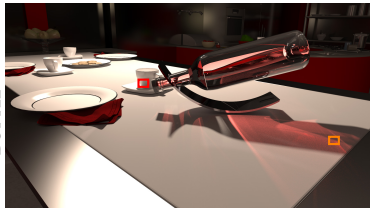
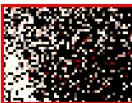
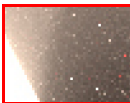
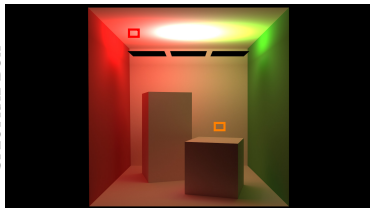
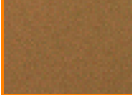
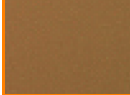
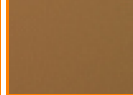
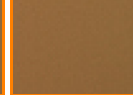
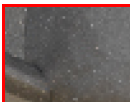
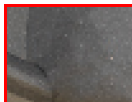
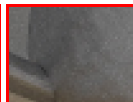
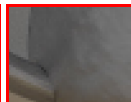
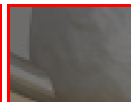
		Unbiased				Biased		
		PT	PPG	NIS	NCV (Ours)	NCV (Ours)	CV Integral	Reference
BATHROOM								
								
	MAPE:	0.112	0.073	0.038	0.033	0.017	0.024	
BEDROOM								
								
	MAPE:	0.064	0.035	0.032	0.029	0.022	0.036	
BOOKSHELF								
								
	MAPE:	0.658	0.045	0.050	0.047	0.032	0.088	
BOTTLE								
								
	MAPE:	0.848	0.088	0.065	0.077	0.053	0.179	
SPECTRAL BOX								
								
	MAPE:	0.030	0.014	0.016	0.010	0.009	0.019	
VEACH DOOR								
								
	MAPE:	0.532	0.084	0.066	0.047	0.028	0.037	

Fig. 7. Neural control variates (NCV) compared to neural importance sampling (NIS) [Müller et al. 2019], practical path guiding with recent improvements (PPG) [Müller 2019; Müller et al. 2017], and uni-directional path tracing (PT) at equal render time. All methods were rendered at a resolution of 1920x1080 pixels for 2h. Unbiased NCV achieves a moderate MAPE improvement in most scenes—the biggest benefit is seen in scenes with smooth, indirect illumination (BATHROOM, SPECTRAL BOX, and VEACH DOOR). We also show two counterexamples where unbiased NCV is outperformed by NIS (BOTTLE) and by PPG (BOOKSHELF). However, when estimating the tail contribution of light transport paths by the learned CV integral driven by our heuristic described in Section 5.1 (biased NCV), we consistently outperform the other techniques (biased NCV column). This variant of NCV results in an additional reduction of noise while introducing only little visible error, 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 1920x1080 pixels and were rendered for 2h. The achieved samples per pixel are written next to the error numbers. Our unbiased neural control variates (NCV) outperform neural importance sampling (NIS) on almost all scenes, except for the ARTROOM and the BOTTLE. The performance advantage is larger when the illumination is spatiodirectionally smooth, such as in interior scenes (e.g. BATHROOM, SPECTRAL BOX, and VEACH DOOR). Using our neural control variate as a light-field oracle after interactions with rough surfaces results in a biased image with further drastically reduced error (biased NCV)—this result implies that the variance reduction outweighs the amount of introduced bias by far. In some scenes (TORUS and VEACH LAMP), lowest MAPE is achieved by using the light-field oracle at the first camera vertex (CV Integral), but the produced images suffer from visually displeasing artifacts (see Figure 7 and the supplementary material).

	PT	Unbiased				Biased		
		[Müller 2019]		[Müller et al. 2019]		Ours		
		PPG		NIS no decay	NIS	NCV	NCV	CV Integral
ARTROOM	1.393 1,756spp	0.108 2,998spp	0.084 1,150spp	0.065 1,181spp	0.081 1,043spp	0.061 1,043spp	0.104 1,043spp	
BATHROOM	0.112 1,785spp	0.073 2,155spp	0.041 667spp	0.038 648spp	0.033 676spp	0.017 676spp	0.024 676spp	
BEDROOM	0.064 1,727spp	0.035 2,187spp	0.034 612spp	0.032 621spp	0.029 636spp	0.022 636spp	0.036 636spp	
BOOKSHELF	0.658 2,113spp	0.045 2,749spp	0.059 868spp	0.050 895spp	0.047 882spp	0.032 882spp	0.088 882spp	
BOTTLE	0.848 2,110spp	0.088 3,612spp	0.089 1,190spp	0.065 1,313spp	0.077 1,108spp	0.053 1,108spp	0.179 1,108spp	
CORNELL BOX	0.035 8,614spp	0.009 3,873spp	0.009 870spp	0.008 883spp	0.006 1,156spp	0.006 1,156spp	0.020 1,156spp	
CRYTEK SPONZA	1.340 1,518spp	0.056 2,417spp	0.066 558spp	0.058 532spp	0.058 650spp	0.046 650spp	0.209 650spp	
GLOSSY KITCHEN	1.450 2,092spp	0.071 2,391spp	0.074 811spp	0.063 841spp	0.059 806spp	0.046 806spp	0.148 806spp	
COUNTRY KITCHEN	0.696 2,070spp	0.068 3,013spp	0.074 946spp	0.071 930spp	0.067 912spp	0.048 912spp	0.077 912spp	
NECKLACE	0.289 10,280spp	0.057 9,449spp	0.041 2,710spp	0.035 2,721spp	0.034 2,760spp	0.032 2,760spp	0.134 2,760spp	
SWIMMING POOL	0.451 4,271spp	0.035 5,771spp	0.042 1,868spp	0.035 1,908spp	0.034 1,858spp	0.031 1,858spp	0.094 1,858spp	
SPACESHIP	0.017 6,489spp	0.009 7,529spp	0.009 2,454spp	0.009 2,563spp	0.009 2,244spp	0.008 2,244spp	0.030 2,244spp	
SPECTRAL BOX	0.030 9,563spp	0.014 4,348spp	0.017 930spp	0.016 951spp	0.010 1,199spp	0.009 1,199spp	0.019 1,199spp	
SPONZA ATRIUM	1.614 1,904spp	0.060 3,007spp	0.050 556spp	0.042 553spp	0.041 726spp	0.023 726spp	0.065 726spp	
STAIRCASE	0.137 1,458spp	0.029 2,553spp	0.024 1,093spp	0.023 1,122spp	0.023 783spp	0.018 783spp	0.036 783spp	
TORUS	0.214 12,108spp	0.021 9,470spp	0.021 3,311spp	0.019 3,211spp	0.016 3,334spp	0.015 3,334spp	0.014 3,334spp	
VEACH DOOR	0.532 3,749spp	0.084 2,773spp	0.072 561spp	0.066 535spp	0.047 682spp	0.028 682spp	0.037 682spp	
VEACH LAMP	0.532 4,079spp	0.069 2,204spp	0.083 499spp	0.069 471spp	0.057 655spp	0.037 655spp	0.025 655spp	

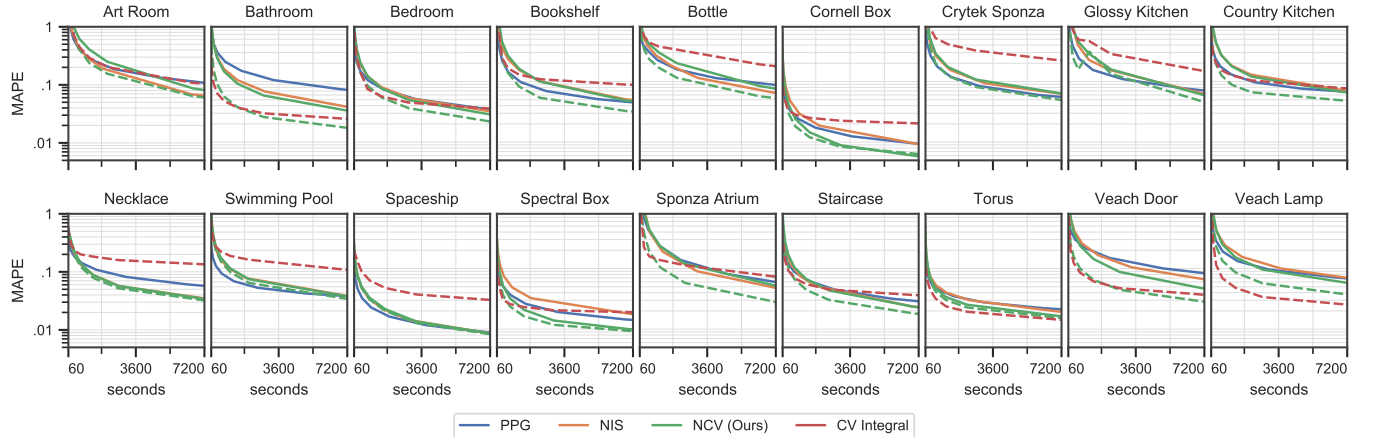


Fig. 8. MAPE convergence plots of practical path guiding (PPG) [Müller et al. 2017], neural importance sampling (NIS) [Müller et al. 2019], and our neural control variates (NCV). The dashed red line corresponds to immediately using the CV integral at the first non-specular vertex. The dashed green line corresponds to the biased variant of our algorithm, where the path suffix is replaced with the learned CV integral using our 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.

6 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 of our technique 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 $2\times$ smaller MAPE loosely corresponds to $4\times$ faster rendering.

Table 2 and Figure 7 summarize our main results. We report MAPE and samples per pixel after 2 hours of rendering at a resolution of 1920×1080 . We compare unidirectional path tracing (PT), practical path guiding with recent improvements (PPG) [Müller 2019; Müller et al. 2017], neural importance sampling (NIS) [Müller et al. 2019], and our neural control variates (NCV). Among these unbiased techniques, our NCVs usually yield the lowest error. We note, that our NCVs often achieve the lower number of samples per pixel. Future performance improvements in neural networks, or more expensive ray-tracing and shading, would therefore improve our results.

Comparison to NIS. To rule out the possibility of our NCVs outperforming NIS simply because it uses additional neural networks for its CV components, we reduced its number of importance-sampling warps (coupling layers) from $L = 4$ to $L = 2$. 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 (5), all with the same architecture, and the same total number of piecewise-quadratic warps (4) as NIS; therefore the number of trainable parameters is the same and the performance is comparable. Differences in samples per pixel are largely caused by differences in importance sampling and thereby path length.

Path termination using NCV. We also show the results of applying our path termination heuristic (Section 5.1) as a by-product of NCVs. The technique dramatically outperforms the unbiased algorithm at the cost of minimally visible artifacts (cf. the “Biased NCV (Ours)” column in Figure 7). Please refer to the supplementary material with an interactive image viewer for full-resolution images.

Quality of our approximations. Table 2 and Figure 7 also contain 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 9. For each scene that we show, 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 only some 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 our supplementary video, which also features the NECKLACE scene as another failure case of incorrectly learned glossy light transport.

Lastly, we demonstrate the *spatial* adaptivity of our neural networks in Figure 10, where we show the learned CV and PDF at several locations in space. Combined with the observations from

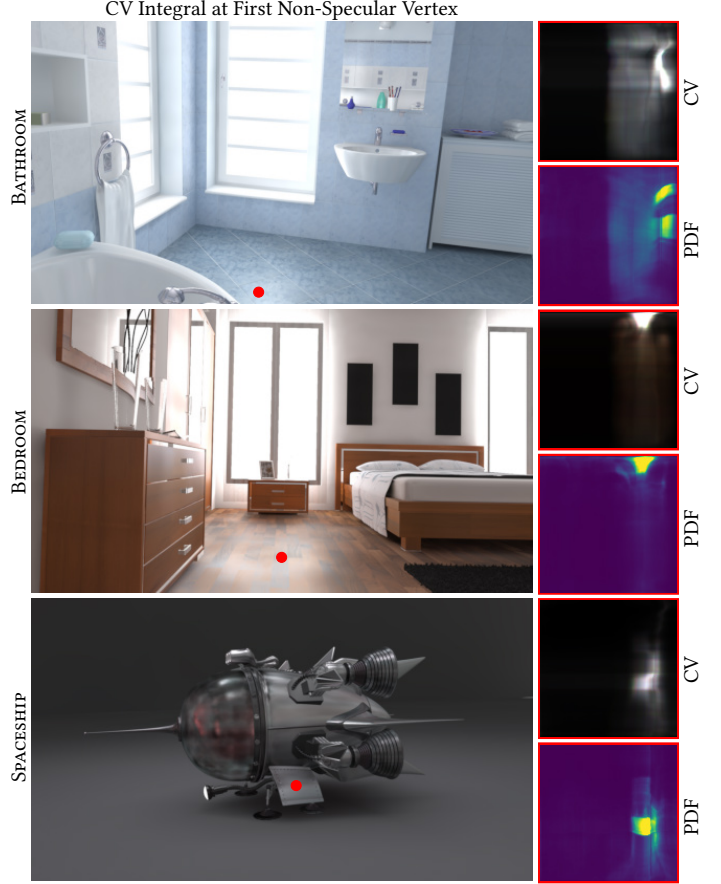


Fig. 9. Visualization of the learned scattered light field (the CV integral G) from novel viewpoints in several scenes. The light field was trained while rendering the corresponding scenes in Figure 7. As the shown camera views were not used for rendering, they were learned by our neural networks from 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).

Figure 9, the spatial adaptivity confirms that the learned CV and PDF capture the full 7-dimensional integrand of the rendering equation with a reasonable accuracy. In our supplementary video, we also visualize the continuous spatial variation of the learned distributions.

Convergence plots. In Figure 8, we plot MAPE vs. time (seconds) for PPG, NIS, and our unbiased and biased NCV applications. Our unbiased NCVs (green line) are mostly on-par or slightly better than NIS, except for the CORNELL BOX and the SPECTRAL BOX, where the difference is more pronounced. Adding our 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 (near-)unbiased techniques overtake the significantly biased learned CV integral in most scenes.

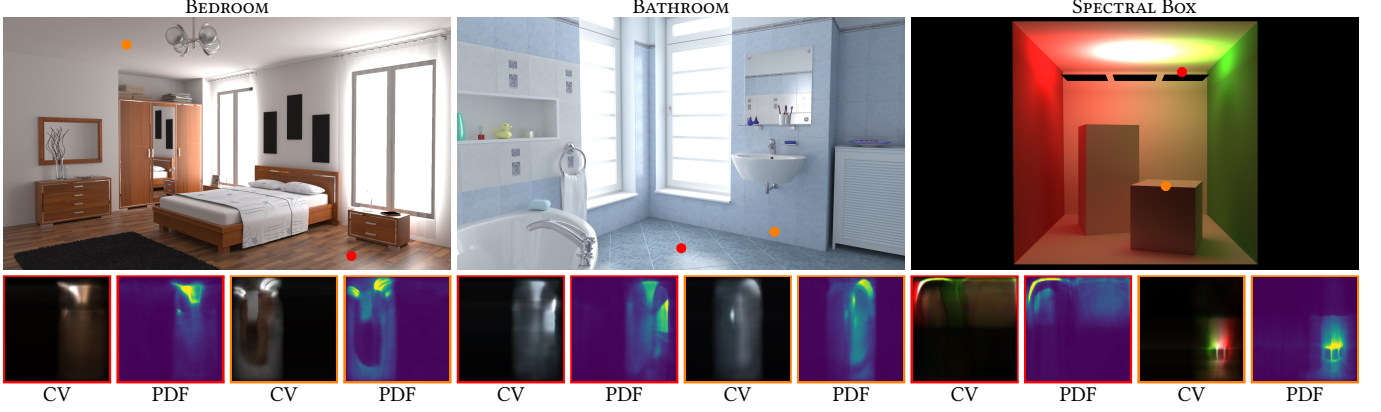


Fig. 10. Visualization of the learned control variates and importance sampling distributions in several scenes. We show CVs (left) and sampling PDFs (right) at two locations (red and orange) per scene. The CVs and PDFs are parametrized in world space via cylindrical coordinates as learned by our neural networks.

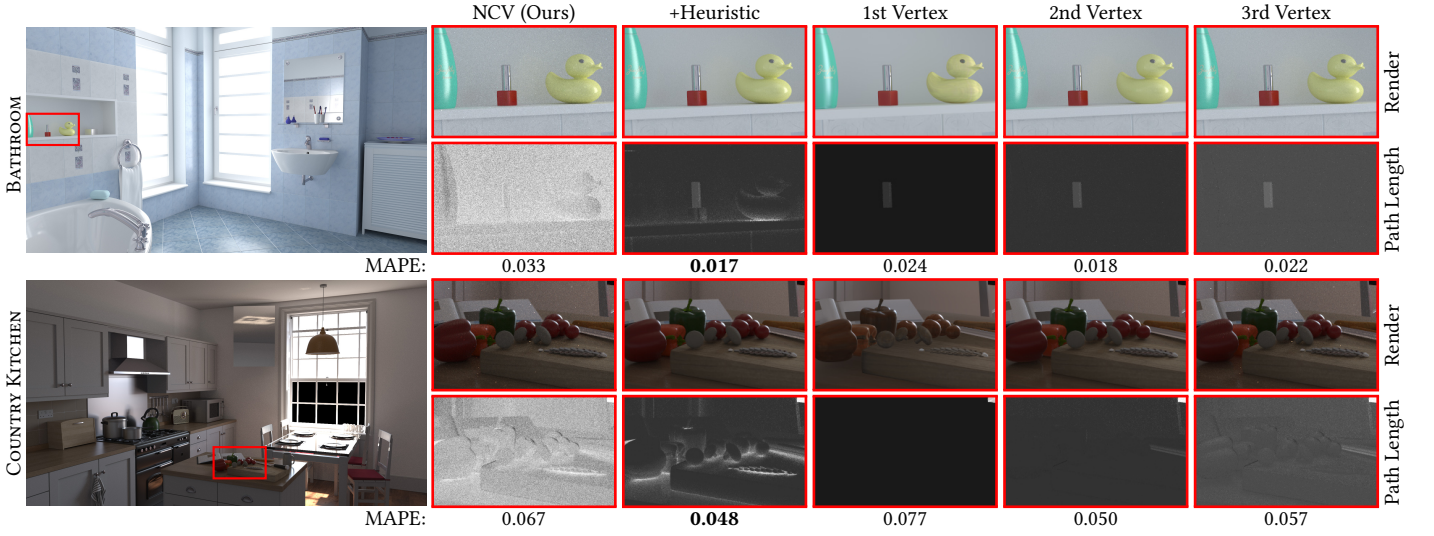


Fig. 11. Different path termination strategies. We compare our unbiased NCVs (no early termination) to our heuristic (Section 5.1) and to terminating at the 1st, 2nd, or 3rd non-specular vertex. The top row shows the resulting images and the bottom row shows the average path length in each pixel (brighter means longer). Our 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).

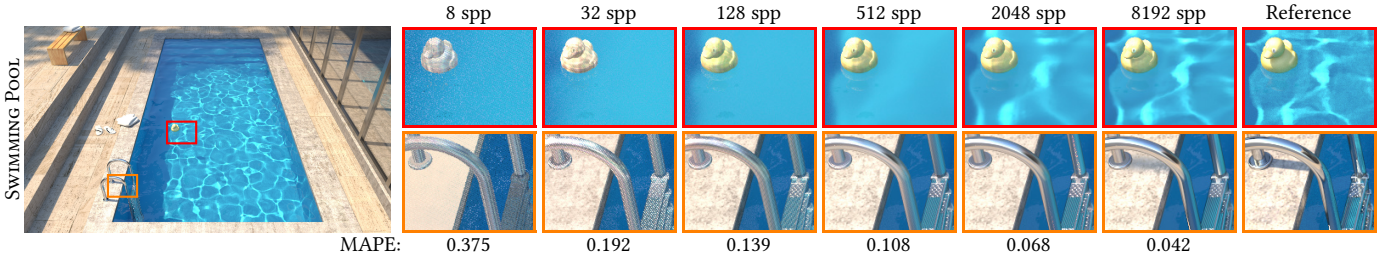


Fig. 12. CV integral training convergence. Training the CV integral for longer than 2h enables more accurate approximations. This suggests that our method is not limited by the approximation power of our neural networks, but by the speed of their training.

7 DISCUSSION AND FUTURE WORK

Separate learning of α and G . Since α and G both are scaling factors of the control variate, at first sight it might seem reasonable to combine them into a single hypothetical variance-minimizing scaling factor $\hat{G}$. The reason we cannot do this is that a variance-minimizing $\hat{G}$ would not necessarily approximate F well, but we need a good approximation $G \approx F$ for in two use cases: (i) in our relative losses that are normalized by G^2 and (ii) our estimate of the tail-contribution of light paths using G .

Handling of signed integrands. In Section 3, we point out that our parametric neural control variates are non-negative by construction. Here, 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:

$$g^+(x) \approx \max(f(x), 0) \quad (37)$$

$$g^-(x) \approx -\min(f(x), 0) \quad (38)$$

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:

$$\begin{aligned} g(x) &\equiv g^+(x) - g^-(x) \\ &\approx \max(f(x), 0) + \min(f(x), 0) = f(x) \end{aligned} \quad (39)$$

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 only little benefit in light-transport simulations (Table 2). 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 light-transport simulation, the integrand is higher dimensional (7D) and less well behaved, leading to a larger distance between our models and a zero-variance configuration. This hypothesis is supported by the fact that simpler scenes benefit more from our control variates, be they mostly spectral (the SPECTRAL BOX scene) or mostly not (the CORNELL BOX scene). Nonetheless, using the learned integral of the control variate (see Section 5.1) results in a huge efficiency gain.

Approximation power. Figure 12 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. For future work, it is therefore of particular interest to investigate means of increasing the training efficiency of the model.

Is NCV practical? The use of neural networks makes our technique more computationally expensive than many non-neural approaches such as PPG (see Table 2) or Gaussian or von Mises-Fischer mixture models [Herholz et al. 2019; Vorba et al. 2014]. Nonetheless, even in a light-weight path tracer such as Mitsuba, the per-sample variance improvements of NCV outweigh the added computational cost and

result in the best overall efficiency. Furthermore, when the cost of shading and tracing rays is relatively high—such as in some production renderers—the relative overhead of our neural networks will be smaller and a larger efficiency attainable. Lastly, when the small amount of bias from our path-termination heuristic is acceptable, the average path length is reduced by roughly 3× (see Figure 11). Efficiency is therefore improved twofold: lower sample variance *and* much cheaper paths, which we did not account for in our results. Developing an algorithm that can harness the cheaper paths—e.g. via Q-learning—may bring neural estimators of light transport closer to interactive and real-time applications. Leveraging samples across time may also become an enabler.

Adjoint Russian roulette and splitting. Many path tracers utilize Russian roulette to terminate paths and thereby probabilistically avoid evaluating low-contribution samples. We did not utilize Russian roulette in our approach under the premise that globally optimal importance sampling would produce samples with equal weight. In practice, the inaccuracies of our importance sampling lead to non-equal weights, which could be corrected using adjoint-driven Russian roulette and splitting (ADRRS) [Vorba and Krivánek 2016]. As Russian roulette also increases variance, the investigation of ADRRS within our parametric control variates is left as future work.

8 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 on average better 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 image synthesis. Our work connects neural light field approximation and unbiased light transport simulation. While data-driven approaches can obtain photorealistic results, correcting their errors is tedious when

only relying on data. Physically based integrators, on the other hand, provide accurate solutions, albeit at excessive computational 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. We believe our work provides the means to correct visual artifacts of seemingly very different approaches, e.g. neural textures [Thies et al. 2019]. 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

We thank Markus Kettunen and Nikolaus Binder for valuable feedback. We also thank the following people for providing scenes and models that appear in our figures: Benedikt Bitterli [2016], Frank Meinel (CRYTEK SPONZA), Jay-Artist (COUNTRY KITCHEN), Johannes Hanika (NECKLACE), Marko Dabrović (SPONZA ATRIUM), Miika Aittala, Samuli Laine, and Jaakko Lehtinen (VEACH DOOR), Olesya Jakob (TORUS), Ondřej Karlík (SWIMMING POOL), SlikDrako (BEDROOM), thecali (SPACESHIP), Tiziano Portenier (BATHROOM, BOOKSHELF), and Wig42 (STAIRCASE).

A VARIANCE OF THE CONTROL VARIATE ESTIMATOR

In order to derive the variance of the control variate estimator

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

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:

$$\begin{aligned} \mathbb{V}[\langle F \rangle] &= \mathbb{V}[\langle F \rangle - \hat{G}(\theta_{\hat{G}})] \\ &= \underbrace{\mathbb{E}[(\langle F \rangle - \hat{G}(\theta_{\hat{G}}))^2]}_{=: U} - \underbrace{\mathbb{E}[\langle F \rangle - \hat{G}(\theta_{\hat{G}})]^2}_{=: V}. \end{aligned} \quad (41)$$

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

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

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

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

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

REFERENCES

- Martin 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>
- 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.
- 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. Monte Carlo and Quasi-Monte Carlo Methods 2006. Springer, Berlin, Heidelberg, Chapter Improved Multilevel Monte Carlo Convergence using the Milstein Scheme, 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. *Monte Carlo and Quasi-Monte Carlo Methods 2000*. Springer, Berlin, Heidelberg, Chapter Some New Perspectives on the Method of Control Variates, 27–49. https://doi.org/10.1007/978-3-642-56046-0_3
- 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. *Advances in Stochastic Simulation Methods*. Birkhäuser Boston, Boston, MA, Chapter The Multilevel Method of Dependent Tests, 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 Krivánek. 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>
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* 2, 5 (July 1989), 359–366. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)
- 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](https://doi.org/10.1016/S0378-4754(00)00248-2) The Second [IMACS] Seminar on Monte Carlo Methods.
- A.G.Z. Kemna and A.C.F. 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](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 Krivánek. 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>
- 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>
- 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 (Proceedings of the 19th Eurographics Symposium on Rendering)* 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*.
- 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> <https://onlinelibrary.wiley.com/doi/pdf/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 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>
- Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph.* 33, 4 (Aug. 2014).
- Jiří Vorba and Jaroslav Krivánek. 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).
- 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>