Recursive Control Variates for Inverse Rendering

BAPTISTE NICOLET, École Polytechnique Fédérale de Lausanne (EPFL) and NVIDIA, Switzerland
FABRICE ROUSSELLE, NVIDIA, Switzerland
JAN NOVÁK, NVIDIA, Czech Republic
ALEXANDER KELLER, NVIDIA, Germany
WENZEL Jakob, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
THOMAS MÜLLER, NVIDIA, Switzerland

Fig. 1. We improve physically based differentiable rendering by introducing a recursive control variate into the optimization loop. Left: a baseline method [Vicini et al. 2021] repeatedly renders noisy images to compute gradients. The noise leads to poor convergence. Right: applied on top of the baseline, our recursive control variate reduces the noise using information from similar renderings in prior optimization steps. This leads to a reconstruction that is much closer to the reference image. The plots show the evolution of the loss function and rendering variance as the optimization progresses.

We present a method for reducing errors—variance and bias—in physically based differentiable rendering (PBDR). Typical applications of PBDR repeatedly render a scene as part of an optimization loop involving gradient descent. The actual change introduced by each gradient descent step is often relatively small, causing a significant degree of redundancy in this computation. We exploit this redundancy by formulating a gradient estimator that employs a recursive control variate, which leverages information from previous optimization steps. The control variate reduces variance in gradients, and, perhaps more importantly, alleviates issues that arise from differentiating loss functions with respect to noisy inputs, a common cause of drift to bad local minima or divergent optimizations. We experimentally evaluate our approach on a variety of path-traced scenes containing surfaces and volumes and observe that primal rendering efficiency improves by a factor of up to 10.

1 INTRODUCTION

Inverse rendering—the reconstruction of a virtual 3D scene from images—has become a mainstream tool for content capture and creation. Differentiable renderers, in which the rendering function is inverted by gradient descent, rely on a costly optimization loop: for each gradient descent step, of which there may be thousands, the algorithm has to render an image of the current state of the reconstruction ("primal" rendering) and compute gradients with respect to scene parameters ("adjoint" rendering). Yet, consecutive gradient descent steps often correspond to minuscule changes in the scene, meaning that a large portion of the rendering computation is redundant.

For this reason, it is common to use only a few samples per pixel, resulting in noisy primal rendering and noisy gradients. After all, optimizers like stochastic gradient descent are designed to deal with noisy gradients, and small noisy steps permit a more fine-grained exploration of the parameter space.

However, a subtle aspect that is often neglected is that the optimizer is not directly applied to the noisy image, but to a loss computed from it. Many common loss functions introduce bias when...
differentiated with respect to noisy inputs, even when the images are themselves unbiased. Affected are all non-$L^2$ loss functions, including the frequently used $L^1$ error. Optimizations using learned perceptual losses such as the widely used VGG loss [Johnson et al. 2016] tend to break down entirely when provided with poorly converged images. Prior work has used tens to hundreds of samples per pixel [Nimier-David et al. 2022, 2019] to alleviate bias-related issues at a considerable computational cost.

Our work leverages the redundancy in the primal rendering phase by using prior renderings as control variates. Control variates [Fieller and Hartley 1954; Kalos and Whitlock 1986] are a generic, unbiased variance reduction technique for Monte Carlo integration. They are particularly appealing in the context of inverse rendering since gradient descent naturally leads to high correlation between subsequent optimization steps, and the noise reduction is proportional to this high correlation.

To this end, we propose a recursive control variate that combines all prior renderings, weighted such that its correlation with the next rendering, and thus the noise reduction, is maximized. To determine the aforementioned weighting, we derive a recurrence relation in terms of the statistics of our estimators, which we in turn estimate with an exponentially weighted variant of Welford’s algorithm [1962].

It is imperative that the derivative estimate has low bias, as it can cause gradient descent to drift to undesirable local minima or even diverge. The recursive scheme overcomes the need for the large number of samples in rendering. In fact, when the optimization reaches steady state, our recursive control variate becomes equivalent to rendering with a correspondingly larger sample count. Compared to prior work, we report up to $10^4$ higher rendering efficiency, leading to either a significant speedup at equal reconstruction quality or higher reconstruction quality after an equal amount of time; see Figure 1.

Our paper starts by reviewing related work (Section 2) followed by the derivation of our recursive control variate (Section 3). Section 4 presents an investigation of the sources of bias in differentiable rendering. We conclude with a thorough evaluation (Section 6) and discussion (Section 7) of our method.

2 RELATED WORK

Our method is related to differentiable rendering, temporal reuse in graphics, and the theory of control variates. The following paragraphs review relevant prior work in all three areas.

**Differentiable rendering.** The term differentiable rendering can refer to a variety of different techniques including rasterization [Hasselgren et al. 2021; Kato et al. 2017; Laine et al. 2020; Liu et al. 2019; Loper and Black 2014; Munkberg et al. 2021; Petersen et al. 2019; Rhodin et al. 2015], sphere tracing of signed distance functions [Bangaru et al. 2022; Vicini et al. 2022], ray marching through emissive volumes [Barron et al. 2021; Mildenhall et al. 2020; Müller et al. 2022; Yu et al. 2021], path tracing of globally illuminated meshes [Azinovic et al. 2019; Bangaru et al. 2020; Hasselgren et al. 2022; Li et al. 2018; Zhang et al. 2020], volume rendering [Che et al. 2020; Gkioulekas et al. 2013; Khungurn et al. 2016; Nimier-David et al. 2022; Zhang et al. 2021], and combinations thereof [Jakob et al. 2022a; Nimier-David et al. 2020, 2019; Vicini et al. 2021; Zeltner et al. 2021].

Our recursive control variates are largely agnostic to the underlying algorithm. They require only minimal changes to reduce noise, e.g., by controlling the pseudo-random number generator seed or fixing path vertices across optimization steps. Thus, our method is orthogonal and beneficial to most of the above baselines—even more so, the noisier they are. In Section 6, we demonstrate higher-quality material optimization with Path Replay Backpropagation (PRB) [Vicini et al. 2021] and twice faster volume optimization with Differential Ratio Tracking (DRT) [Nimier-David et al. 2022].

**Temporal computation reuse.** Practitioners have long sought to amortize the rendering cost across consecutive frames both in interactive applications and in offline rendering of animated sequences. Existing work ranges from biased screen-space approaches like temporal anti-aliasing [Karais 2014; Marrs et al. 2018] and denoising [Chaitanya et al. 2017; Hasselgren et al. 2020; Schied et al. 2018; Vogels et al. 2018], over scene-space data structures like photon mapping [Elek et al. 2012] and illumination caches [Binder et al. 2022; Majercik et al. 2022; Müller et al. 2021; Seyb et al. 2020] to unbiased techniques such as path guiding [Dittebrandt et al. 2020], reservoir resampling [Bitterli et al. 2020; Lin et al. 2022; Talbot et al. 2005], and control variates [Keller 2001; Manzi et al. 2016; Rousselle et al. 2016].

Two properties of control variates are particularly appealing in the context of differentiable rendering: they are simple to implement and provide unbounded noise reduction when the optimization reaches a steady state, which equivalent to rendering with an increasingly larger sample count. Ease of implementation implies that they can sometimes even be applied on top of prior work. We demonstrate a proof-of-concept combination with the OptiX denoiser [Chaitanya et al. 2017] in Figure 9.

Pertaining to differentiable rendering, concurrent work in cloud tomography [Czerninski and Schechner 2023] resamples paths across consecutive optimization steps in order to avoid the computational cost of tracing new ones. However, the authors report that this comes at the cost of increased noise—a reversal of the characteristics of our method.

**Control variates.** Training pipelines in the area of machine learning tend to rely on variants of stochastic gradient descent (SGD) to replace the costly gradient evaluation with a cheaper statistical estimate computed from a randomly selected subset of the training data. The resulting variance can impede convergence, which has motivated variance-reduced methods [Gower et al. 2020] including Stochastic Average Gradient “Améliore” (SAGA) [Defazio et al. 2014] and Stochastic Variance-Reduced Gradient (SVRG) [Johnson and Zhang 2013]. Both SAGA and SVRG maintain a running approximation of the full gradient and use it to provide unbiased variance reduction using the framework of control variates. Such approaches are not compatible with differentiable rendering, where variance arises in a different manner owing to the numerical integration over continuous domains.

To reduce rendering noise, the first use of control variates is due to Lafortune and Willems [1995a,b] who fit a constant ambient lighting term and a 5D piecewise-constant tree to the spatio-directional radiance field. Because these fitted functions are analytically and cheaply integrable, they are trivially usable as control variates.
3 VARIANCE REDUCTION BY CORRELATED SAMPLING

We introduce a new generalization of control variates [Fieller and Hartley 1954] that facilitates its adoption in optimization algorithms as an error reduction technique.

3.1 Preliminaries

Control variates. We use the notation $\langle F \rangle$ to denote a Monte Carlo estimator of an integral $F$ involving the function $f$. This estimator may exhibit a large amount of variance. Given another function $g$ resembling $f$, whose integral $G$ is furthermore available in closed form, the variance of estimating $F$ can be reduced by the addition of a control variate [Fieller and Hartley 1954]

$$\langle F \rangle_{CV} = \langle F \rangle + \alpha (G - \langle G \rangle). \tag{1}$$

For this technique to be effective, the estimator $\langle G \rangle$ should be correlated with $\langle F \rangle$, which is typically accomplished by evaluating both estimators using the same set of pseudo-random numbers.

Since $\mathbb{E}[G - \langle G \rangle] = 0$, the real-valued control weight $\alpha$ may be chosen arbitrarily without introducing bias. The optimal choice that minimizes variance [Owen 2013] is given by

$$\alpha_{\text{opt}} = \frac{\text{Cov} \{ \langle F \rangle, \langle G \rangle \}}{\text{Var} \{ \langle G \rangle \}}. \tag{2}$$

When $G$ is not known in closed form, a Monte Carlo estimate can be used instead, which we denote by $\langle G \rangle_0$. The variance of such an estimator is minimized by the optimal control weight

$$\alpha_{\text{opt}} = \frac{\text{Cov} \{ \langle F \rangle, (G) - \langle G \rangle_0 \}}{\text{Var} \{ (G) - \langle G \rangle_0 \}}. \tag{3}$$

Building on Equation (3), Rousselle et al. [2016] show how to reduce the variance when re-rendering an existing scene following a change to its parameters. Let $\langle F \rangle_p$ denote an initial rendering computed using a particular set of Monte Carlo samples with index $p$, while $\langle F \rangle_q$ refers to the estimator following a modification of the scene. If $p = q$, the second estimator uses the same sample set, causing it to be correlated with $\langle F \rangle_p$. If $p \neq q$, then $p$ and $q$ will be deemed statistically independent sets of samples. In general, $\langle F \rangle_p$ will refer to a rendering of an evolving scene at iteration $i$ using the set of samples $p$.

Rousselle et al. [2016] propose to use the original rendering $\langle F \rangle_0$ as a control variate, resulting in the estimator

$$\langle F \rangle_{CV} = \langle F \rangle_1 + \alpha (\langle F \rangle_0 - \langle F \rangle_1), \tag{4}$$

where the weight

$$\alpha = \frac{\text{Cov} \{ \langle F \rangle_1, \langle F \rangle_0 \}}{\text{Var} \{ \langle F \rangle_0 - \langle F \rangle_1 \}} \tag{5}$$

is computed using numerical approximations of the (co-)variances. The estimator $\langle F \rangle_0$ does not appear in the covariance (cf. Equation (3)) because it is statistically independent from $\langle F \rangle_1$ and therefore does not contribute.

3.2 Recursive Control Variates

Applications of differentiable rendering tend to re-render the scene thousands of times. At the same time, the magnitude of changes between steps is relatively small—often orders of magnitude below those found in primal renderings of animated or interactive content. Our extension of the previously introduced control variate exploits this redundancy across an arbitrarily long sequence of renderings. We generalize the concept in Equation (4) to a sequence, where the new estimator

$$\langle F_n \rangle_{CV} = \langle F_n \rangle_n + \alpha_n (\langle F_{n-1} \rangle_{CV} - \langle F_{n-1} \rangle_n)$$

recursively compounds variance reduction across the entire sequence.
While the structure of the denominator above does not match Equation (8), we instead rely on Welford’s algorithm [1962], which computes an equivalent estimate in a single pass using a fixed amount of temporary storage. Appendix A reviews the original form of Welford’s algorithm along with exponential weighting [Schubert and Gertz 2018; Xie and Olano 2021] and a modification to avoid startup bias. The estimates no longer identically distributed. We use an exponentially weighted aggregation to compromise between the two goals. The estimates become unbiased once the optimization reaches a steady state.

The control weight \( \alpha_n \) has an intuitive interpretation: it determines the degree to which past information can be reused. Large scene parameter changes will generally reduce the correlation between current and previous renderings, causing \( \alpha_n \) to be nearly zero. This effectively discards the information gathered in previous iterations. Conversely, an unchanging scene is maximally correlated, in which case the numerator equals \( \text{Var}[\langle F_{n-1} \rangle_n] \). If all renderings are maximally correlated, we have \( \alpha_n = n/n+1 \). Inserting this expression into Equation (6) reveals that \( \langle F_{i} \rangle_{CV} \) turns into the mean of all preceding estimates. Figure 2 visualizes the per-pixel control weights during an inverse rendering optimization. In early iterations, the control weights start out close to zero, progressively approaching unity as more and more information is gathered. The variance of the rendered primal images decreases correspondingly.

**Optimality.** While the approach we just presented uses all previous renderings to reduce variance at a given iteration, it does not combine them in the theoretically optimal way, which would be to use all previous renderings as individual control variates:

\[
\langle F_n \rangle_{CV} = \langle F_n \rangle_n + \sum_{i=0}^{n-1} \alpha_i (\langle F_i \rangle_i - \langle F_i \rangle_n)
\]  

(9)

However, such an approach is not practical: determination of control weights \( \alpha_i \) entails rendering all previous states using the same set of Monte Carlo samples to construct and process an ever-growing covariance matrix. As we observed, earlier terms become less important as the optimization evolves, hence this prohibitive computational effort would also be wasteful. Our method can be interpreted as an approximation of this strategy, whose weights are equivalent to Equation (9) when the optimization is in its steady state, as illustrated in Figure 3.

**Estimating the control weight \( \alpha_n \).** The control variate described so far remains incomplete in the sense that it still lacks an explanation of how the (co-) variance terms in Equation (8) should be computed. Unfortunately, these terms are intractable using analytic methods, which means that approximations are needed in practice. The better these approximations, the closer the control variate will approach theoretical optimality.

Our method estimates the needed terms \( \text{Cov}[\langle F_n \rangle_n, \langle F_{n-1} \rangle_n] \), \( \text{Var}[\langle F_{n-1} \rangle_n] \), and \( \text{Var}[\langle F_{n-1} \rangle_{CV}] \) statistically, hence we seek approximations that are characterized by a small amount of variance and bias. Compared to other parts of the optimization, some bias in the statistics estimation is tolerable, since it merely impacts the effectiveness of the control variate.

Low variance and low bias are somewhat conflicting goals: the use of samples from many prior optimization steps will naturally produce more converged estimates. At the same time, long-term aggregation with an evolving scene introduces bias, as samples are no longer identically distributed. We use an exponentially weighted aggregation to compromise between the two goals. The estimates become unbiased once the optimization reaches a steady state.

The most commonly used (co-) variance estimator performs two passes through a dataset to determine the sample mean and final estimate. This is problematic in long-running optimizations, since every evaluation of \( \alpha_n \) would need access to all prior renderings. We instead rely on Welford’s algorithm [1962], which computes an equivalent estimate in a single pass using a fixed amount of temporary storage. Appendix A reviews the original form of Welford’s algorithm along with exponential weighting [Schubert and Gertz 2018; Xie and Olano 2021] and a modification to avoid startup bias.

A final challenge arises when estimating the variance of \( \langle F_n \rangle_{CV} \): the recursive definition of our control variate introduces correlations across iterations, which violates independence assumptions in standard variance estimators. We propose a recurrence that accounts for this effect:

\[
\text{Var}[\langle F_n \rangle_{CV}] = \text{Var}[\langle F_n \rangle_n] + \alpha_n^2 \text{Var}[\langle F_{n-1} \rangle_n] - 2 \alpha_n \text{Cov}[\langle F_n \rangle_n, \langle F_{n-1} \rangle_n] + \alpha_n^2 \text{Var}[\langle F_{n-1} \rangle_{CV}]
\]  

(10)
Although correct, there is a practical issue with this equation: exact statistics involving \((\langle F_0 \rangle)\) and \((\langle F_{n-1} \rangle)\) are generally unknown.

We found that the recurrence remains effective when it is used in an approximate sense, by estimating statistics with the presented variant of Welford’s algorithm. Figure 3 examines the effectiveness of this approximation in a setting where optimal weights can be computed analytically.

Although the control weight can in principle take on any value, e.g., \(\alpha_n < 0\) to exploit negative correlation, such a situation would not be meaningful in our application that repeatedly renders the same scene with small variations. We always clamp \(\alpha_n\) to the range \([0, 1]\), which is helpful especially at the beginning of the optimization when the statistics are not yet well-converged.

Bias. Unbiasedness of a control variate generally requires that its input estimators are not correlated with the control weight [Lavender et al. 1982; Nelson 1990]. This is not the case in our method, which is a key limitation requiring further discussion. Indeed, our initial definition of \(\alpha_n\) correlates it with respect to both \((\langle F_{n-1} \rangle)_{CV}\) and \((\langle F_{n-1} \rangle)_n\), which leads to noticeable bias shown in Figure 4 (left).

In practice, we introduce a small modification by updating the running (co-)variance estimates after evaluating \((\langle F_n \rangle)_{CV}\). The statistics estimators thus lag one iteration behind, ensuring that \(\alpha_n\) and \((\langle F_{n-1} \rangle)_n\) are statistically independent. Some bias clearly remains, as \(\alpha_n\) is still correlated with \((\langle F_{n-1} \rangle)_{CV}\) (both computed using a stream of previous renderings). In other words, our approach removes the correlation with the previous rendering, but not with all the ones before it. The right side of Figure 4 illustrates the impact of this change. As we will show in the next section, our method can significantly reduce the amount of bias caused by loss function derivatives. As such, we trade a small amount of bias in the control variate for a larger bias reduction in the loss function gradient.

4 BIAS IN DIFFERENTIABLE RENDERING

Derivative-based inverse rendering methods require some type of loss function \(L\) to quantify the difference between the simulation’s output \(F_0\) and reference data \(I_{ref}\). Different representations of \(F_0\) and \(I_{ref}\) are conceivable: in the simplest case, this could be a single image. More commonly, multiple images from different viewpoints must be combined to reduce ambiguity. Instead of an image, the renderer could also output a sparse sampling of the light field or a projection onto a set of basis functions. The observations in this section apply to all of these possibilities.

We usually seek the global minimizer \(\theta^*\) of the composition of simulation and loss, i.e.,

\[
\theta^* = \arg \min_{\theta} L(F_0, I_{ref}).
\]

Due to the challenging nature of this non-linear objective, we must taper our expectations and settle for a local minimum. This normally involves gradient-based techniques based on the derivative of the objective. Following expansion via the chain rule, an estimator of this derivative is given by

\[
\langle \partial_\theta L(F_0, I_{ref}) \rangle = \langle \partial_\theta F_0 \rangle \cdot \partial L((F_0), I_{ref}).
\]

A subtle issue with this approach is that the estimate can be biased even when \((\langle F_0 \rangle)\) is unbiased, which breaks traditional convergence guarantees of stochastic gradient descent (SGD). For example, SGD with averaging finds the minimum of any strongly convex objective, but it may drift or even diverge when fed with biased gradients. A first potential source of bias arises from the multiplication operation in Equation (12), which is only unbiased if both factors are independent estimators. The established way to deal with this constraint is to simply estimate \((\langle F_0 \rangle)\) twice with different random variables [Gkioulekas et al. 2013]. We use path replay back-propagation (PRB) [Vinci et al. 2021] to estimate the first factor \((\partial_\theta F_0)_{PRB}\) and insert our recursive control variate \((\langle F_0 \rangle)_{CV}\) into the loss term. This results in the final form of the proposed gradient estimator:

\[
\langle \partial_\theta L(F_0, I_{ref}) \rangle = \langle \partial_\theta F_0 \rangle_{PRB} \cdot \partial L((\langle F_0 \rangle)_{CV}, I_{ref}) .
\]

The second source of bias is less well studied and harder to overcome: \(\partial L((\langle F_0 \rangle), I_{ref})\) is only unbiased when \((\langle F_0 \rangle)\) is noise-free, or when the derivative of the loss \(\partial L\) is a linear function, since \(\mathbb{E}(f(X)) \neq \mathbb{E}(f(X))\) for general random variables \(X\) unless the function \(f\) is linear.2 Here, \(f\) corresponds to \(\partial L\) and \(X\) to \((\langle F_0 \rangle)\).

This observation has dire consequences: in general, the \(L^2\) loss is the only loss function that leads to unbiased derivative estimates. While this may be an acceptable choice for some applications [Mildenhall et al. 2020; Mülller et al. 2021], many other situations call for different loss functions [Hasselgren et al. 2020; Heitz et al. 2021; Johnson et al. 2016; Kallweit et al. 2017; Munkberg et al. 2021; Xing et al. 2022].

1For the \(L^2\) loss specifically, there exist alternative solutions [Deng et al. 2022; Pidhorzkyi et al. 2022] that not only produce unbiased gradients but also unbiased loss values (though the values do not affect optimization, see Figure 6). In our work, we argue for supporting different loss functions so we do not discuss these solutions in further detail.

2For convex \(f\), this is also known as Jensen’s inequality [1906].
The control variate in Equation (6) references the terms \( \langle F_n \rangle_n \) and \( \langle F_{n-1} \rangle_n \), which represent correlated renderings of the scene at adjacent iterations performed using the same Monte Carlo samples. This still leaves some room for experimentation—for example, the preceding sections did not state the domain on which samples should be generated. We tested sample reuse on primary sample space [Kelemen et al. 2002] and on path space [Veach 1997]. Figure 8 visualizes the impact of this choice on the control variate’s performance.

**Primary sample space.** Sample reuse on primary sample space is particularly simple: all one needs to do is to render the scene twice at every iteration. The sample generator should be initialized so that its internal state in every first rendering step matches that of the preceding iteration’s second rendering step. We denote this approach as “CV-PSS” in our experiments.

There are two downsides: first, rendering twice at every iteration naturally doubles the primal rendering cost, which accounts for roughly \(1/3\) of the total cost of a PRB gradient step.

The second issue is the way in which certain scene parameters influence the sampled path geometry. For example, consider sampling the directional distribution of a rough microfacet BRDF using a fixed position in primary sample space. Perturbing the roughness parameter will also change the computed ray direction, which can cascade into a significant change to the rest of the light path. The resulting loss of correlation is apparent in Figure 8 (b). Parameters with this behavior include surface roughness, normal maps, the index of refraction, and the extinction and scattering anisotropy of participating media.

**Path space.** The path-space variant reuses sampled light paths instead of the uniform variates that were used to generate them. The architecture of our implementation (“CV-PS”) follows Rousselle et al. [2016], who introduce BSDF adapters that encapsulate two BSDFs with different parameter states. We extend this idea to media and rely on a modified volumetric path tracer that samples paths from the new state, while simultaneously computing radiance estimates for the old state. Participating media furthermore require that free-flight distances are sampled with a majorant that bounds both states. Appendix B provides pseudocode explaining the necessary modifications in the surface case.

\[
\langle \partial L(F_0) \rangle_{BFD} = \langle \partial L(F_0) \rangle + \partial L(\langle F_0 \rangle) - \mathbb{E}[\partial L(F_0)].
\]
We implemented our method in Mitsuba 3 [Jakob et al. 2022b] and combined it with two distinct differentiable rendering techniques: Path Replay Backpropagation (PRB) [Vicini et al. 2021] to reconstruct textured surface materials and homogeneous volume parameters, and Differential Ratio Tracking (DRT) [Nimier-David et al. 2022] to reconstruct heterogeneous volumes. We construct a separate control weight per color channel. Figures that visualize control weights show the ones for the red channel. Our reference implementation can be found at https://github.com/rgl-epfl/recursive_control_variates.

Fig. 7. Our control variate reduces both variance and bias in typical reconstruction tasks. Here, we contrast loss function derivatives of the baseline (PRB) and our method computed using either a $L^1$ or $L^2$ loss as loss function. While our method cannot fully remove all bias in the $L^1$ case, the output still shows good agreement with reference derivatives computed using a high sample count. The example shows iteration #20 of an optimization that reconstructs the materials of a challenging interior scene lit entirely using indirect illumination, where all approaches use the same gradient updates, i.e. control variates are only used for visualization. Further detail on the optimization is provided in Section 6 and Figure 9.

Fig. 8. Our method builds on the ability to re-render a changing scene using the same set of Monte Carlo samples. The three columns show renderings of the scene in Figure 2 at taking the same 50 optimization steps. They compare (a) The baseline without control variate. (b) Sample reuse in primary space ("CV-PS"). While this approach is easy to implement by re-seeding the sample generator, it can suffer from reduced correlation when scene parameters impact the sampled path geometry. This is apparent in the optimization of bunny #2’s roughness and bunny #3’s extinction parameter. (c) Sample reuse in path space ("CV-PS") leads to an overall higher correlation, control weight values, and lower variance.

It is interesting to note that rendering algorithms already evaluate correlated estimators in path space, using a single path to estimate the radiance for each of several color channels. Another architectural approach for realizing path-space sample reuse could therefore entail changing the system’s color representation from 3 (RGB) to 6 channels or tracking two packets of sampled wavelengths.

Path space sample reuse yields improved correlation and lower variance. Reusing light paths to evaluate two estimators also reduces ray tracing cost and makes it possible to fuse the entire computation into a single GPU kernel. Applied to the experiment in Figure 8, we find that this approach reduces primal rendering overheads from 2× to 1.6× (which, again, only represents roughly 1/5 of the total cost of a gradient step). Besides Figure 8, all other results discussed in this paper use the CV-PS variant.

Geometric optimization. We did not investigate parameters that move or deform surface geometry, which pose additional challenges for sample reuse. In the primary sample space setting, we expect such parameters to reduce the correlation and effectiveness of the control variate similar to other parameters that have an effect on the sampled path geometry. The path space variant would require the addition of “shift mappings” to ensure that the moved path vertices remain valid [Manzi et al. 2016]. We find this an interesting direction for future work but consider it beyond the scope of this article.

Volumetric representations replace discrete visibility changes at boundaries with smooth transitions. They are less affected by these challenges and lend themselves to gradually evolving geometry.

6 RESULTS

We implemented our method in Mitsuba 3 [Jakob et al. 2022b] and combined it with two distinct differentiable rendering techniques: Path Replay Backpropagation (PRB) [Vicini et al. 2021] to reconstruct textured surface materials and homogeneous volume parameters, and Differential Ratio Tracking (DRT) [Nimier-David et al. 2022] to reconstruct heterogeneous volumes. We construct a separate control weight per color channel. Figures that visualize control weights show the ones for the red channel. Our reference implementation can be found at https://github.com/rgl-epfl/recursive_control_variates.

Material reconstruction. The inversion task in Figure 9 analyzes the performance of material reconstruction in a scene with challenging indirect illumination. We hold the geometry fixed and reset the painting’s texture and teapot materials. The optimization then tries to recover these properties, as well as homogeneous volume parameters for the teapot. We use the gradient preconditioning approach of [Nicolet et al. 2021] for the texture gradients. We use PRB [Vicini et al. 2021] as baseline and experimented with $L^1$ and $L^2$ losses that compare the tentative state to a single rendered reference image. Combining PRB with our recursive control variate results in a clear improvement irrespective of the chosen loss function. The improvement is largest when minimizing the $L^1$ loss due to the bias reduction afforded by our technique.

Denoising. Most real-world use of Monte Carlo rendering nowadays involves a dedicated denoising step, which is typically carried out by convolutional neural networks trained on large datasets. It is tempting to incorporate a similar denoising step to reduce primal rendering variance in an inverse rendering pipeline.

Several insets in Figure 9 demonstrate the consequences of this idea. Counter-intuitively, a direct application of a denoiser actually reduces the reconstruction quality, since the added bias impedes the optimization. This can turn a previously unbiased optimization into one that no longer converges— observe, e.g., the unrealistic brightening of the tea in the $L^2$ PRB+Denoising column.

Our experiment used the OptiX denoiser that is based on an article by Chaitanya et al. [2017], but we expect similar behavior with other learned denoisers. Denoising our control variate estimator progressively reduces bias and yields the overall best result.
VGG [Simonyan and Zisserman 2015] is a widely used convolutional neural network that has been trained to classify images from the ImageNet dataset with high accuracy. The related VGG loss [Johnson et al. 2016] truncates VGG and computes the Euclidean distance between high-level features produced during the model’s evaluation.

Figure 9. Reconstruction performance in a challenging scene that is entirely lit using indirect illumination. We reset the painting’s texture and teapot materials and attempt to reconstruct them using a single rendered reference image using Path Replay Backpropagation [Vicini et al. 2021]. Our control variate (“Ours” columns) improves gradients, which leads to higher-quality reconstructions. The improvement consists of reduced variance when using a $L^1$ loss and additional bias reduction when using a $L^2$ loss and/or denoiser. Counter-intuitively, the addition of a denoising step can reduce reconstruction quality: while it lowers primal rendering noise, the added bias tends to impede the optimization. Applying the denoiser on top of our control variate estimator yields better results.

Fig. 10. Learned perceptual losses are sensitive to noise, which can bias the optimization towards suboptimal solutions. Here, we employ a VGG loss [Johnson et al. 2016] to quantify the distance to the reference (c) in a reconstruction of the back wall’s albedo texture. The PRB baseline (a) is visibly affected by loss-induced bias, while our method in (b) yields a better match.

Neural loss. Learned neural loss functions play an increasingly important role as a differentiable proxy for perceptual similarity. VGG [Simonyan and Zisserman 2015] is a widely used convolutional neural network that has been trained to classify images from the ImageNet dataset with high accuracy. The related VGG loss [Johnson et al. 2016] truncates VGG and computes the Euclidean distance between high-level features produced during the model’s evaluation.

Such techniques present an interesting alternative to traditional pixel-wise losses, but they also tend to be highly sensitive to noise due to the deep features propagating it.

The optimization in Figure 10 shows this by attempting to reconstruct the textured back wall of a Cornell box, as measured by VGG feature distance to a reference. Column (a) shows that loss-induced bias is a severe problem, while our control variate in (b) achieves a superior match to the reference in (c).

Heterogeneous volume reconstruction. Figure 11 applies our technique to the problem of heterogeneous volume reconstruction building on the adjoint pass of DRT. We modified the original implementation provided by the authors so that it processes full images instead of randomly sampled subsets, which our implementation does not currently support. We consistently use the same 8-10 reference views for each reconstruction task. Experiments in the original article [Nimier-David et al. 2022] relied on a $L^1$ loss, which made it necessary to render primal images with a relatively large sample count to reduce the impact of loss function bias. Variance reduction from the control variate alleviates the need for this high sample count, which improves the reconstruction quality at equal time and significantly reduces primal rendering overheads for when targeting an equal level of quality. Detailed inspection of these results is available through our supplementary image viewer.

7 DISCUSSION

Control variates for the adjoint phase. The recursive control variate as presented in Section 3 is not limited to primal rendering—in principle, the technique applies to any sequence of Monte Carlo estimators. It should therefore be possible to also apply it to another conspicuous sequence of Monte Carlo estimators that occurs in the inverse rendering pipeline, namely the adjoint rendering phase represented by the left factor in Equation (12) that transforms the loss gradient into a scene parameter gradient.

However, correlated adjoint rendering is more involved than the sample reuse techniques described in Section 5. Furthermore, the adjoint estimator does not share the bias-related issues encountered in the primal phase. Modern optimizers such as Adam [Kingma and Ba 2015] compute accurate updates from unbiased gradients, even if they are very noisy, which makes the potential benefits uncertain. We leave further investigation of this topic as a future work.

Heuristic control weights. Our results suggest that accurate reconstruction and low variance go hand in hand with large values of the control weight $\alpha_n$, e.g. in Figures 2 and 8. It is tempting to encourage such positive outcomes heuristically by setting $\alpha_n = 1$ or, informed by the steady-state solution, $\alpha_n = n/n+1$, neither of which depend on estimated statistics.
However, both heuristics have clear failure modes. The first results in unbounded variance because the previous control variate is never down-weighted while independent random variation continues to be added per Equation (6). The second heuristic results in a uniform combination over all prior control variates regardless of how similar each one is to the current scene. The optimal value of $\alpha_n$ strongly depends on the optimization trajectory, which motivates our definition of $\alpha_n$ in Section 3.2.

**Sample reuse in volumes.** Introducing correlation in heterogeneous volume rendering by reusing sampled path vertices can increase the cost of rendering. Delta tracking [Woodcock et al. 1965] requires the majorant to bound both old and new volumes, which adds overhead in the form of increased numbers of null collisions. The density change from one iteration to the next fortunately tends to be relatively small, which limits this additional cost. Similar to the observation regarding color channels in Section 5, rendering a volume with two states resembles rendering a volume with spectrally varying extinction and albedo. Thus, our method may also benefit from more sophisticated rendering techniques for spectrally varying volumes [Kutz et al. 2017; Miller et al. 2019].

**Spatial reuse.** Our method exploits redundancy by reusing information from prior optimization steps. Spatial reuse presents another optimization opportunity, as demonstrated in gradient-domain path tracing [Kettunen et al. 2015] and image space control variates [Keller 2001; Rousselle et al. 2016]. Curiously, recent inverse rendering techniques decrease spatial redundancy following the observation that processing randomly chosen pixels from all viewpoints improves convergence [Mildenhall et al. 2020; Nimier-David et al. 2022], as opposed to traditional rendering of coherent rays emanating from a single viewpoint. Future techniques targeting reuse across space and time will need to address such extra complications.
Multi-level Monte Carlo. Besides their theoretical optimality [Heinrich 1998], hierarchical Monte Carlo methods and especially telescopic sums have become a tool for debiasing estimators [Misso et al. 2022]. Hence there are two opportunities for future research: exploring the efficiency of a hierarchical approach [Keller 2001] and investigating in how far biased gradient estimators can be made at least consistent by means of multi-level Monte Carlo methods.

Combination with other variance reduction techniques. In the same spirit as the combination with denoising in Figure 9, we believe other, orthogonal, variance reduction techniques exploiting temporal, screen space, and scene space correlation, are applicable in tandem with our approach. Promising candidates are temporal path guiding [Dittebrandt et al. 2020] and reservoir resampling [Bitterli et al. 2020], which are both effective at spatio-temporal information reuse, are both unbiased, and are both computationally light weight.

8 CONCLUSION
Practically all contemporary physically based rendering techniques reuse information across simulated light paths in some form in order to be efficient, be that temporally or spatially. Differentiable rendering introduces an additional, as-yet untapped treasure trove of optimization reuse: the optimization loop. Our recursive control variate is a first attempt at leveraging this redundancy and we expect many more to be practical.

What makes our technique powerful is its generality. Section 3 applies to any Monte Carlo estimator that is used in conjunction with gradient descent, such as in simulations of quantum systems. Debiasing loss derivatives, as well as in developing further schemes of potential information reuse, are both unbiased, and are both computationally light weight.

ACKNOWLEDGMENTS
The authors would like to thank the reviewers for their insightful feedback, Tizian Zeltner for helpful discussions in the early stages of the paper, Valentin Deschaintre for discussions on the VGG loss, and Aaron Lefohn for his support. Volumes, environment maps and 3D models were generously provided by JangaFX, PolyHaven and vajrable. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No 948846).

REFERENCES

ACM Trans. Graph., Vol. 42, No. 4, Article 1. Publication date: August 2023.
A EXponentially Weighted Welford’s Algorithm

The control variate in Section 3.2 depends on variance and covariance estimates that we compute via Welford’s algorithm [1962], which requires a single pass and a fixed amount of temporary memory. One additional requirement from Section 3.2 is that the influence of prior states decays according to an exponentially weighted moving average (EWMA). The iteration should furthermore be initialized to remove startup bias. We have not found a variant combining these three aspects in prior work and therefore describe it here.

\[ \mu_n = \alpha \mu_{n-1} + (1 - \alpha)x_n \]
\[ \mu_n = (1 - \alpha) \sum_{k=0}^{\infty} \alpha^k x_{n-k}. \]

However, an infinite history is not available in practice, which means that the sum turns finite and begins with a first observation \( x_0 \):

\[ \mu_n = (1 - \alpha) \sum_{k=0}^{n} \alpha^k x_{n-k}. \]

Suppose that all elements \( x_n \) are in fact constant, i.e., \( x_n = x_0 \). In this case, it would be desirable that the moving average agrees with this value, but summation of the geometric progression produces

\[ \mu_n = (1 - \alpha^{n+1}) x_0. \]

To account for the finite past window of observations, we must scale the EWMA values by \( (1 - \alpha^{n+1})^{-1} \). Another way of avoiding truncation error is to assign the weight 1 to the first sample by setting \( \mu_0 := x_0 \). This leads to the following alternative EWMA:

\[ \tilde{\mu}_n = \alpha^n x_0 + (1 - \alpha) \sum_{k=0}^{n-1} \alpha^k x_{n-k}, \]

which is equivalent to infinite-extent EWMA on the stream:

\[ \tilde{x}_n = \begin{cases} x_0 & n \leq 0, \\ x_n & n > 0. \end{cases} \]
Consequently, the estimator \( \hat{\mu}_n \) is unbiased when used to compute an average of iid random variables, but its output is initially skewed towards \( x_0 \), as shown in Figure 12. This is undesirable.

```
def update_covariance(C, Mx, My, x, y, n):
    # Update the means
    My += dy / n
    Mx += (x - Mx) / n
    # Update the second moment
    C += dy * dx
    # Return the sample covariance
    return C / (n - 1)
```

Listing 1. Original code for Welford’s online algorithm for estimating the covariance of two random variables.

Welford EWMA. We found prior use of weighted variants of Welford’s algorithm work [Schubert and Gertz 2018; Schubert et al. 2014, 2016; Xie and Olano 2021], but none used the former variant of start-up bias correction described above.

The following snippet realizes a Welford EWMA for covariance estimation. Estimating the variance can be achieved in a similar manner. The example uses Python syntax somewhat leniently and assumes that an assignment to an argument propagates to the caller. We use two differential decay rates \( \alpha = 0.9 \) and \( \beta = 0.999 \) to average first and second-order statistics following Kingma and Ba [2015]. The unmodified code is provided in Listing 1.

```
def update_covariance(C, Mx, My, x, y, \alpha, \beta, n):
    # Substitute the bias-corrected mean
    dy = y - My
    dy *= \alpha**n
    Mx *= \alpha**n
    # Substitute the bias-corrected mean
    dx = x - Mx
    dx *= \alpha**n
    # Update the second moment
    C *= \beta**n
    # Return the bias-corrected covariance
    return C / (1 - \beta**n)
```

Listing 2. Modified version of Welford’s algorithm to use exponential moving averages to compute covariances.

B IMPLEMENTING PATH-SPACE CORRELATED RENDERING.

In order to introduce correlation in path space, one needs to evaluate two states of the scene at the same time. We achieve this by augmenting a standard path tracer by a few more function calls to evaluate the previous state of the scene while tracing. We show a simplified example for a surface path tracer below.

```
def sample(scene, ray):
    L_old, L_new = 0
    mis_em = |
    mis_dir = |
    if prev_bsdf_delta:
        mis_dir = mis_weight(...) |
        L_old /= \beta * mis_dir * si.emission |
        L_new /= \beta * mis_dir * si.emission
    # Emitter sampling
    wo, em_weight = scene.sample_emitter()
    val = si.bsdf.eval(wo)
    val_old = si.bsdf.eval_old(wo)
    mis_em = mis_weight(...) |
    L_old /= \beta * mis_em * val_old * em_weight |
    L_new /= \beta * mis_em * val * em_weight
    # BSDF sampling
    wo, val, pdf = si.bsdf.sample()
    # Evaluate the current path for the previous state
    val_old = si.bsdf.eval_old(wo)
    prev_bsdf_delta = si.bsdf.is_delta()
    if prev_bsdf_delta:
        mis_em = |
        mis_dir = |
        L_old /= \beta
        else:
            mis_em = |
            mis_dir = |
            L_old /= \beta
```

Listing 3. Example of the modifications for a basic surface integrator with MIS. We assume that the BSDFs have been augmented to keep track of the previous parameter state, which can be queried via bsdf.eval_old.