Optimizing Path Termination for Radiance Caching Through Explicit Variance Trading

LUKAS KANDLBINDER*, Karlsruhe Institute of Technology, Germany
ADDIS DITTEBRANDT*, Karlsruhe Institute of Technology, Germany
ALEXANDER SCHIPEK, Karlsruhe Institute of Technology, Germany
CARSTEN DACHSBACHER, Karlsruhe Institute of Technology, Germany

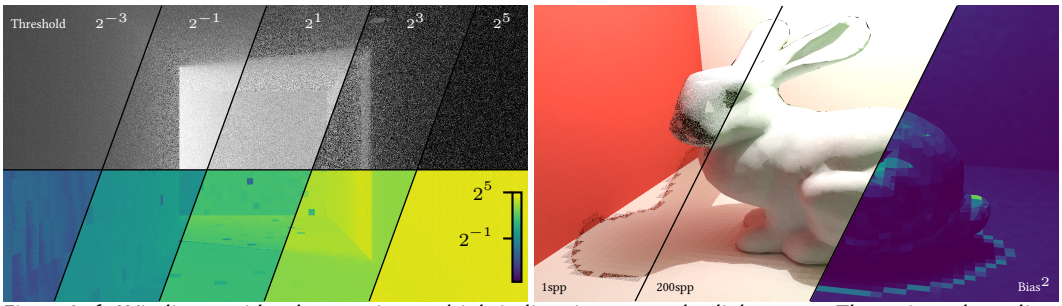


Fig. 1. Left: Winding corridor that requires multiple indirections to reach a light source. The variance bounding scheme is applied with increasing thresholds to allow for longer paths while bounding image variance (visualized below). Right: Bunny in a Cornell box that casts a shadow to the floor. The mean squared error minimization scheme detects cache records with high bias (right visualization) and avoids direct termination.

Radiance caching allows to amortize the cost of path tracing by sharing contributions of path suffices in a spatial data structure. This sharing generally introduces bias, but it can be traded with variance by terminating into the cache at deeper path vertices. We develop a framework to implicitly reduce bias by optimizing path termination towards a chosen variance bound. Importantly, this bound can be chosen large enough while still being amenable to denoising. This results in longer paths being sampled with unbiased path tracing as permitted by the estimator variance and variance bound. To that end, we reformulate the variance of a path tracing estimator as a quantity that can be expressed locally for a path, relying on auxiliary statistics that are shared through the radiance cache structure independently of the path prefix. This allows to perform the optimization locally during path construction, while still translating to a global bound. Our method is capable of maintaining the variance bound in complex scenes, resulting in lower bias compared to other techniques such as heuristics based on ray differentials. We additionally present first findings on directly optimizing the bias-variance tradeoff based on local bias estimates of individual cache records, although the optimization is only approximate, resulting in suboptimal termination decisions.

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

*Both authors contributed equally to the paper.

Authors' Contact Information: Lukas Kandlbinder, Karlsruhe Institute of Technology, Karlsruhe, Germany, lukas.kandlbinder@student.kit.edu; Addis Dittebrandt, Karlsruhe Institute of Technology, Karlsruhe, Germany, addis.dittebrandt@kit.edu; Alexander Schipek, Karlsruhe Institute of Technology, Karlsruhe, Germany, addis.dittebrandt@kit.edu; Carsten Dachsbacher, Karlsruhe Institute of Technology, Karlsruhe, Germany, dachsbacher@kit.edu.

ACM 2577-6193/2024/7-ART
https://doi.org/10.1145/3675381

Additional Key Words and Phrases: Radiance Caching

ACM Reference Format:
Lukas Kandlbinder, Addis Dittebrandt, Alexander Schipek, and Carsten Dachsbacher. 2024. Optimizing Path Termination for Radiance Caching Through Explicit Variance Trading. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 3 (July 2024), 19 pages. https://doi.org/10.1145/3675381

1 Introduction

Photorealistic image synthesis based on light transport simulation provides a unified framework to handle a wide range of lighting effects. Monte Carlo integration techniques such as path tracing [Kajiya 1986] in particular are used extensively in movie production [Droske et al. 2023]. With the advent of hardware ray tracing, Monte Carlo techniques are also starting to be adopted on the GPU for real-time rendering. However, the inherent noise due to the stochastic sampling remains an issue. Furthermore, the cost of generating individual paths is considerable due to incoherent memory accesses. Most applications therefore make only targeted use of path tracing to reproduce specific effects, such as soft shadows or diffuse global illumination.

Radiance caching [Krivanek et al. 2005] allows to amortize the cost of generating paths by storing cache records in a data structure that can be queried for surface points. Instead of tracing full paths per pixel, paths can be terminated early with the stored cache records used to estimate radiance of the suffix. The method is biased, however, as cache records do not store the exact radiance of a given surface point, but an average for surface points associated with a cache record. To hide bias artifacts, cache records are usually not queried at primary vertices but at deeper path vertices dictated by some heuristic.

In this paper, we explicitly optimize radiance cache termination with two separate strategies: (1) Variance bounding termination (section 4.2) that bounds image variance to a configurable threshold. While bias is not explicitly accounted for, cache termination is deferred as much as possible under the variance constraint to minimize the effects of bias. An example scene of a corridor is shown in fig. 1 where the variance bound is varied, resulting in a corresponding image variance. (2) Approximate mean squared error minimizing termination (section 4.3) that, compared to variance bounding, additionally accounts for bias. This approach allows to terminate paths earlier if a cache record represents its contained surface points well enough. Figure 1 right shows the Stanford bunny, where shadow edges are classified as regions with higher bias, forcing the path tracer to continue traversal. Both approaches build on the same princple to perform a global optimization (bounding variance or minimizing MSE) through local termination decisions. We first reformulate both variance and bias as terms that can be optimized locally for a path (section 4.1), allowing for evaluation inline while path tracing. We also show how to estimate variance and bias in a cache record by tracking three stochastic moments of the path tracing estimator (section 4.4), two of which are the first and second moment and one additional marginal second moment that distinguishes between variance and bias.

2 Related Work

Radiance caching. The approach to amortize the cost of path tracing by storing cache records in a world-accessible data structure dates back to irradiance caching [Ward et al. 1988] which only considers diffuse interreflections. It was later extended with directional information to represent actual radiance [Krivanek et al. 2005]. In classic radiance caching literature, termination is performed directly at primary vertices. Therefore, many works focus on optimizing the representation to reduce visible error [Krivanek et al. 2005; Marco et al. 2018; Schwarzhaupt et al. 2012]. Other approaches use simpler representations but hide artifacts by deferring evaluation to deeper path vertices. Path space filtering [Keller et al. 2016] defers evaluation always to the second path vertex,

which resembles final gathering. Neural radiance caching [Müller et al. 2021] uses a heuristic based on an approximate area-spread [Bekaert et al. 2003] of paths, that quantifies how much paths diverge. We compare against the area-spread heuristic in section 6.

Various cache representations were developed, such as hashed grids [Binder et al. 2019; Hachisuka and Jensen 2010], texture space [Munkberg et al. 2016], surfels [Halen et al. 2021], cards [Wright et al. 2022] and neural [Hadadan et al. 2021; Müller et al. 2022, 2021]. Our approach is largely orthogonal to those, since we only add additional quantities to the cache record. The exception are neural representations, as cache records are usually encoded opaquely in a latent vector space, making a direct application nontrivial. For simplicity, we implement our approach using hashed grids only.

Our variance estimates necessitate an iterative update scheme that is comparable to recent works [Deng et al. 2021; Pantaleoni 2020]. It transports radiance directly between caches instead of estimating radiance through path suffices.

Photon mapping. Lighting effects involving specular-diffuse-specular paths are notoriously difficult to solve with path tracing. Photon Mapping [Jensen 1996] approximately solves this by distributing photon particles initiated from light paths across the scene and then estimate indirect illumination at a surface point using kernel density estimation. Final gathering is often applied to hide artifacts emerging from kernel density estimation by deferring photon map access after a diffuse interaction. Alternatively, the kernel width can be optimized, which is known as bandwidth selection in the wider literature and has been investigated in the context of photon mapping [Kaplanyan and Dachsbacher 2013]. It is a tradeoff between bias in the form of blurring or variance due to an insufficient amount of photon particles. While radiance caching can be framed similarly, the mechanism by which we trade bias with variance (deferred path termination) needs to be formalized differently.

Adaptive sampling. Initially introduced to reduce edge aliasing in image space [Mitchell 1987], adaptive sampling allocates more sampling budget to challenging parts of the image or scene [Hachisuka et al. 2008]. Controlled path termination can also be interpreted as an adaptive sampling technique, since early termination of paths has a similar effect.

Variance estimation. Estimates of variance or second moment are also used for other purposes, for example, to steer guiding [Rath et al. 2020], control russian roulette and splitting factors [Rath et al. 2022], or blurring radii for denoising [Schied et al. 2017].

Denoising. Raw path tracer output is usually not presented directly to the user due to disturbing noise artifacts. It is instead fed into a post-processing pipeline to remove residual noise through controlled blurring. Classic approaches include edge-avoiding à-trous wavelets [Schied et al. 2017] and neural techniques [Chaitanya et al. 2017; Thomas et al. 2022]. Our algorithms, variance bounding in particular, aim to provide output with predictable (bounded) noise characteristics.

3 Background

Path integral. We derive our termination strategies using the path integral formulation [Veach 1998]. Instead of defining light transport as a recursive integral [Kajiya 1986], it can be reformulated as a flat integral $I = \int_\Omega f(\mathbf{X})\,\mathrm{d}\mathbf{X}$ in area-product measure Ω on the measurement contribution function

$$f(\mathbf{X}) = W(x_0)G(x_0, x_1)\left(\prod_{i=1}^{k-2} f_r(x_{i-1}, x_i, x_{i+1})G(x_i, x_{i+1})\right) L_e\left(x_{k-2}, x_{k-1}\right),$$

which is the product of sensor responsivity W, a series of geometry terms G and BRDFs f_r and emission L_e. Given a probability density function p on the same area-product measure, a corresponding Monte Carlo estimator $g = f(\mathbf{X})/p(\mathbf{X})$ can be defined. We also define a split version of $g = g_l^{\text{pre}} g_l^{\text{suf}}$, which splits the terms of g before and after vertex l, with $0 < l < k - 1$:

$$g_l^{\text{pre}} = \frac{W(x_0)G(x_0, x_1)\left(\prod_{i=1}^{l-1} f_r(x_{i-1}, x_i, x_{i+1})G(x_i, x_{i+1})\right)}{p\left(x_0, \ldots, x_l\right)},$$

$$g_l^{\text{suf}} = \frac{\left(\prod_{i=l}^{k-2} f_r(x_{i-1}, x_i, x_{i+1})G(x_i, x_{i+1})\right) L_e\left(x_{k-2}, x_{k-1}\right)}{p\left(x_{l+1}, \ldots, x_k \mid x_0, \ldots, x_l\right)}.$$

Radiance caching. We define a radiance caching estimator as

$$\bar{g} = g_{l(\mathbf{X})}^{\text{pre}} \text{rc}\left(x_{l(\mathbf{X})}, x_{l(\mathbf{X})-1} - x_{l(\mathbf{X})}\right).$$

The estimator borrows a prefix of length $l(\mathbf{X})$ from the original estimator g but estimates the remaining suffix at $x_{l(\mathbf{X})}$ through the radiance cache, expressed through $\text{rc}(x, \omega)$. x defines the spatial location and ω the outgoing direction. To allow for individual per-path decisions, $l(\mathbf{X})$ is a function of the path itself. In practice, terminating at vertex $x_{l(\mathbf{X})}$ should only depend on the prefix $\left(x_0, \ldots, x_{l(\mathbf{X})}\right)$ to allow for incremental construction.

For a real implementation of $\text{rc}(x, \omega)$, it is impossible to maintain accurate radiance estimates between all vertex pairs. We instead maintain statistics for spatio-directional partitions stored in discrete cache records to recover (weighted) averages of radiance. We define a partition as $(x_0, x_1) \in P \subseteq A^2$ (a subset of all vertex pairs). We define the radiance cache as the expectation over the suffix estimator for all vertex pairs in the same partition:

$$\forall (x, y) \in P : \text{rc}(x, y - x) = E_{(x_0, x_1) \in P}[E[g_1^{\text{suf}} \mid x_0, x_1]] = E_P[g_1^{\text{suf}}].$$

The occurrence probabilities of x_0 and x_1 (the joint probability of all path prefixes leading to these vertices) are not accounted for in this formulation, because they are intractable to compute. This means that this expectation gives higher weight to vertex pairs with higher occurrence probability.

Mean squared error (MSE). The mean squared error of an estimator Y is defined as the expected squared difference of its realizations to a reference value $\hat{Y}$. It can be decomposed into the variance of the estimator and the squared bias with respect to the reference:

$$\text{MSE}(Y, \hat{Y}) = V[Y] + \text{Bias}(Y, \hat{Y})^2.$$

We are usually not interested in the MSE of an estimator directly, but rather the MSE of the sample average of N realizations of it. The extended definition of MSE to an N-sample estimator is given as

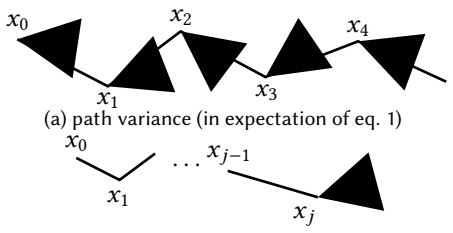
$$\text{MSE}\left(\sum_N \frac{Y}{N}, \hat{Y}\right) = \frac{V[Y]}{N} + \text{Bias}(Y, \hat{Y})^2.$$

Only the variance vanishes when increasing the sample count. Thus, the impact of bias becomes more relevant, since it does not vanish.

4 Method

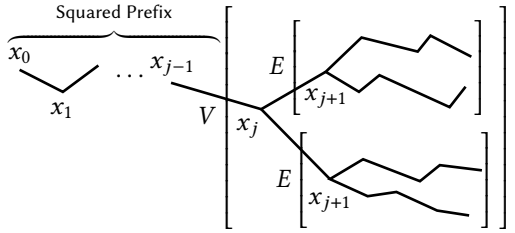
In the following, we derive algorithms to optimize radiance cache termination $l(\mathbf{X})$ based on screen-space variance and Mean-Squared Error (MSE). We do so using termination decisions that are entirely local to a path. To this end, we exploit monotonicity of the expectation operator:

$$X \leq Y \Rightarrow E[X] \leq E[Y].$$



(a) path variance (in expectation of eq. 1)

(b) prefixed local variance (symbolic)



(c) prefixed local variance (summand in expectation of eq. 1)

Fig. 2. (a) Path variance is composed of (b) local variances of stochastic decisions at each path vertex with its prefix. (c) Local variance is defined over a marginalized suffix estimator, accounting only for the next path segment.

An inequality that holds between two random variables X and Y also holds in the expectation. By reformulating variance and MSE as expectations over path space (section 4.1), we can apply this reasoning: If we locally bound or minimize the inner term of the expectation, the expectation itself, and therefore the per-pixel variance and MSE, will also be bounded or minimized. In sections 4.2 and 4.3 we derive algorithms based on the reformulation to bound variance and minimize MSE on a per-pixel basis, respectively. Finally, in section 4.4 we show how to estimate the needed quantities for the aforementioned algorithms.

4.1 Reformulating variance & MSE

Reformulating variance. We initially reformulate the variance for the path tracing estimator g and later relate this expression to the radiance caching estimator $\bar{g}$. We proceed as follows: We leverage the law of total variance:

$$V\left[g\right] = VE\left[g \mid x_1\right] + EV\left[g \mid x_1\right],$$

and apply it recursively by conditioning on successive path vertices:

$$V\left[g\right] = VE\left[g \mid x_1\right] + E\Big[V\Big[E\left[g \mid x_1x_2\right] \;\Big|\; x_1\Big] + E\Big[V\Big[E\left[g \mid x_1x_2x_3\right] \;\Big|\; x_1x_2\Big] + \dots\Big]\Big].$$

Previous work [Bolin and Meyer 1997] decomposes the recursion into variance terms for each level of indirection:

$$V\left[g\right] = VE\left[g \mid x_1\right] + EV\Big[E\left[g \mid x_1x_2\right] \;\Big|\; x_1\Big] + EV\Big[E\left[g \mid x_1x_2x_3\right] \;\Big|\; x_1x_2\Big] + \dots \quad .$$

We move all expectations out of the recursion instead:

$$V\left[g\right] = E\left[\sum_{j=0}^{\infty} V\Big[E\left[g \mid x_1 \dots x_{j+1}\right] \;\Big|\; x_1 \dots x_j\Big]\right].$$

Note that the range $x_1 \dots x_j$ is empty for $j = 0$. This expression is already in the form of an expectation over path space, as required for the local optimization. We perform one additional step: The terms that only depend on conditioned vertices inside the variance can be factored as a squared prefix g_j^{pre} with the suffix g_j^{suf} remaining in the variance term:

$$V\left[g\right] = E\left[\sum_{j=0}^{\infty} \left(g_j^{\text{pre}}\right)^2 \underbrace{VE\left[g_j^{\text{suf}} \;\middle|\; x_{j+1}\right]}_{\text{Local variance (estimation in section 4.4)}}\right] = E\left[\sum_{j=0}^{\infty} P_j[g]\right] \tag{1}$$

Due to the conditional expectation, the variance term expresses the local variance at vertex x_j. This is not to be confused with the variance of the suffix estimator. We refer to the combined term of squared prefix and local variance as prefixed local variance P_l. While the squared prefix can be readily computed for a path, the local variance needs to be estimated separately (section 4.4). We refer to the sum of prefixed local variances as path variance (fig. 2).

Comparing g to $\bar{g}$, it can be observed that $\bar{g}$ is a truncated version of g. This translates analogously to the variance, where the number of sum terms is bound by $l(\mathbf{X})$:

$$V[\bar{g}] = E\left[\sum_{j=0}^{l(\mathbf{X})-1} P_j[\bar{g}]\right]. \tag{2}$$

Reformulating bias. The bias of $\bar{g}$ is defined with respect to the original estimator g:

$$\text{Bias}(\bar{g}, E[g])^2 = (E[\bar{g}] - E[g])^2 .$$

We can factor out the common prefix of both estimators to retrieve a form that relates the bias to the local biases introduced through caching:

$$\text{Bias}(\bar{g}, E[g])^2 = E\left[g^{\text{pre}}_{l(\mathbf{X})}\left(\text{rc}\left(x_{l(\mathbf{X})-1}, x_{l(\mathbf{X})}\right) - E\left[g^{\text{suf}}_{l(\mathbf{X})}\right]\right)\right]^2.$$

This form is still not useful because the outer expectation is squared, so we bound this term from above using Jensen's inequality [Jensen 1906] and move the square into the expectation:

$$\text{Bias}(\bar{g}, E[g])^2 \leq E\Big[g^{\text{pre}}_{l(\mathbf{X})} \underbrace{\left(\text{rc}\left(x_{l(\mathbf{X})-1}, x_{l(\mathbf{X})}\right) - E\left[g^{\text{suf}}_{l(\mathbf{X})}\right]\right)^2}_{\text{Local bias (estimation in section 4.4)}}\Big] = E\left[B_{l(\mathbf{X})}[\bar{g}]^2\right]. \tag{3}$$

This approximation neglects cases where biases of different paths cancel each other out, since this effect is hard to determine when considering just individual paths. Similar to local variance, the local bias also needs to be estimated separately (section 4.4). In the following, we reference this bias term as $B_{l(\mathbf{X})}$.

Reformulating MSE. Combining the reformulated variance and bias (eqs. 2 and 3) gives the reformulated MSE:

$$\text{MSE}\left(\sum_N \frac{\bar{g}}{N}, E[g]\right) = \frac{V[\bar{g}]}{N} + \text{Bias}(\bar{g}, E[g])^2 \leq \frac{1}{N} E\left[\sum_{j=0}^{l(\mathbf{X})-1} P_j[\bar{g}]\right] + E\left[B_{l(\mathbf{X})}[\bar{g}]^2\right].$$

Note that the reformulated MSE is an upper bound of the true MSE due to the approximate bias term. The respective expectations can be joined to receive an expression that can be evaluated individually for each path:

$$\text{MSE}\left(\sum_N \frac{\bar{g}}{N}, E[g]\right) \leq E\left[\sum_{j=0}^{l(\mathbf{X})-1} \frac{P_j[\bar{g}]}{N} + B_{l(\mathbf{X})}[\bar{g}]^2\right] \tag{4}$$

4.2 Variance bounding termination

The goal of the variance-bounding termination strategy is to terminate paths such that the resulting image variance is bounded. The key idea is that the reformulated variance in eq. 2 allows us to perform this bounding locally for individual paths while still translating to a global bound for the

```
1  fn trace_path(pos, dir, thres, target, margin)
2      thp ← 1; con ← 0;
3      path_var ← 0; alpha ← 1;
4      pos ← trace_ray(pos, dir); // terminate if no hit
5      loop
6          pl_var ← thp² · get_local_variance(pos, −dir);
7          if path_var + pl_var > thres then
8              alpha ← √((thres−path_var)/pl_var); // eq. 5
9          pl_bias ← thp² · get_local_bias_sq(pos, −dir);
10         pd_bias ← thp² · get_deferred_bias_sq(pos, −dir);
11         pd_bias ← min (0.9 · margin · pl_bias, pd_bias);
12         c_mse ← path_var/target + pl_bias;
13         d_mse ← (path_var + pl_var)/target + pd_bias;
14         if margin · c_mse < d_mse then
15             alpha ← 0;
16         con ← con + (1 − alpha) · thp · rc(pos, −dir);
17         con ← con + alpha · next_event_estimation(pos, −dir);   // accounting for interpolation
18         dir, weight ← sample_bsdf(pos, −dir);
19         thp ← thp · weight;
20         pos ← trace_ray(pos, dir);
21         if alpha < 1 then                                       // terminate into next cache
22             con ← con + alpha · thp · rc(pos, −dir);
23             break;
24         path_var ← path_var + pl_var;
25     return con;
```

Variable Descriptions	
Shared	
pos	current surface position
dir	current incoming direction
thp	path throughput
con	path contribution
path_var	path variance
alpha	interpolation weight 1 = path tracer, 0 = current cache
pl_var	prefixed local variance
Variance bounding	
thres	absolute variance threshold
MSE minimization	
target	sample target to optimize MSE for
margin	margin factor
pl_bias	prefixed local squared bias
pd_bias	prefixed deferred squared bias
c_mse	current MSE when terminating at current surface
d_mse	deferred MSE when adding one more segment

Fig. 3. Basic path tracer with next event estimation augmented with controlled radiance cache termination (highlighted in green). Blue background: Variance bounding termination with interpolation (section 4.2). Red background: MSE minimizing termination (section 4.3).

image. $l(\mathbf{X})$ can be defined directly based on the path variance:

$$l(\mathbf{X}) = \arg\max_{l \in \mathbb{N}} \left(\sum_{j=0}^{l-1} P_j[\bar{g}] \leq t \right) = \arg\max_{l \in \mathbb{N}} \sum_{j=0}^{l-1} \left(g_j^{\text{pre}} \right)^2 VE\left[g_j^{\text{suf}} \,\middle|\, x_{j+1} \right] \leq t.$$

This means that the termination depth is maximized under the variance constraint.

Variance interpolation. Termination into the cache is performed even if the prefixed local variance P_j of the next vertex would increase the sum just slightly beyond the threshold t. For such cases it is sensible to blend the contribution of the cache with the contribution of one additional segment using an alpha factor α:

$$\sum_{j=0}^{l(\mathbf{X})-1} P_j[g] + P_{l(\mathbf{X})}[\alpha g] = \sum_{j=0}^{l(\mathbf{X})-1} P_j[g] + \alpha^2 P_{l(\mathbf{X})}[g] = t.$$

The additional $P_{l(\mathbf{X})}$ is scaled by α^2. We can solve this expression for α:

$$\alpha = \sqrt{\frac{t - \sum_{j=0}^{l(\mathbf{X})-1} P_j[g]}{P_{l(\mathbf{X})}[g]}}. \tag{5}$$

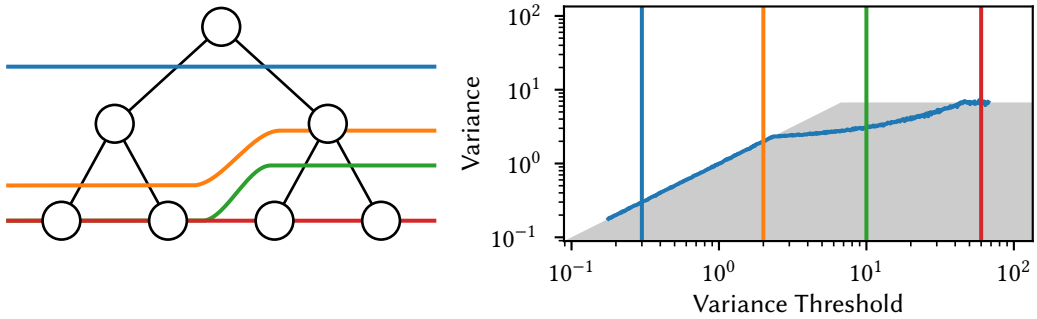


Fig. 4. Discrete test case of variance bounding path tracer implementation (fig. 3). Left: Sampling tree consisting of leaf nodes that emit a constant energy of one unit. Left subtrees are sampled with 90% probability. Right: MSE of path tracer sampling the tree depending on the variance threshold. Colored lines: Termination of path tracer depending on variance threshold, which can be between a node and its children due to interpolation.

Path tracer integration. We have sketched out a basic path tracer implementation with next event estimation in fig. 3 that incrementally constructs a path by maintaining the current surface point (pos, dir) as well as the path throughput (thp) and contribution (con). The code with blue background depicts the variance bounding strategy. The path variance (path_var) can be constructed incrementally by adding the prefixed local variances (pl_var) of each sampled vertex to a cumulative sum. Path termination is decided in line 7 using the variance threshold (thres). When performing interpolation, the computed alpha (alpha) term also needs to be incorporated into any direct illumination (line 17) in addition to the reflected radiance at the next vertex as given by the cache.

Validation. To validate the theory, we have implemented the variance bounding strategy to estimate the contribution of a discrete tree (fig. 4). The leaf nodes emit one unit of energy, which is transported along the edges towards the root node. Left edges are selected with 90% probability. The discrete nature of the test case allows to compute the total reflected energy and local variances of each node in closed form. The plot shows the estimated root mean squared error depending on the variance threshold. We can see that the MSE initially closely follows the threshold. But eventually the variance detaches from the threshold and then converges to a maximum value. The latter happens because the underlying estimator has finite variance, i.e. no path termination takes place. The former is due to the left subtree contributing less variance, thus being fully expanded earlier. Since each path tries to match the variance threshold individually, the MSE will be smaller than the threshold as soon as some subtrees are completely expanded.

4.3 MSE minimizing termination

The MSE minimizing termination strategy (fig. 3, red background) strives to minimize the error in the image for a given number of samples to accumulate. We use the reformulated MSE (eq. 4) and minimize the inner part of the expectation on a per-path basis. Due to incremental path construction, only information associated with the path prefix can be used. Therefore, minimization is performed in a greedy fashion: Before continuing a path, the current MSE (c_mse, line 12) is compared (line 14) against the deferred MSE (d_mse, line 13), which includes an estimate of average bias when terminating at the next vertex (pd_bias)

Heuristic factors. In part due to the bounded bias term, the aforementioned comparison is invariant of the prefix - most terms cancel, including the path throughput. This means that termination is entirely dependent on the local statistics of a cache record. To counter this, we introduce a margin

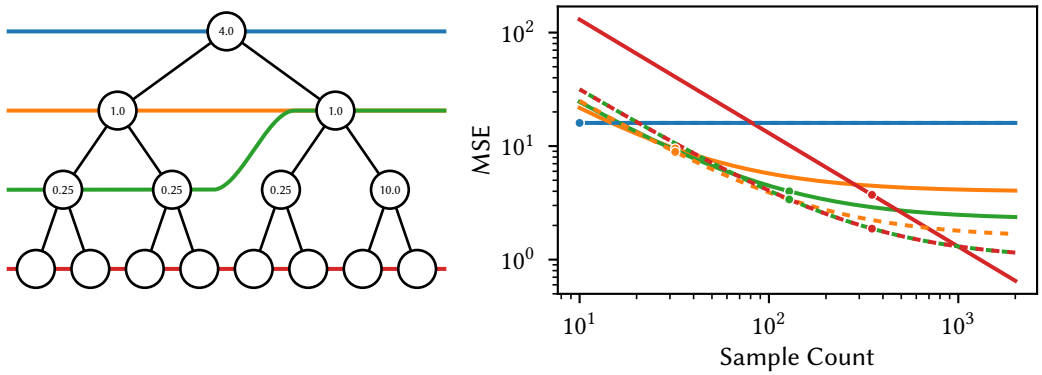


Fig. 5. Discrete test case of MSE minimizing path tracer implementation (fig. 3). Left: Sampling tree consisting of leaf nodes that emit a constant energy of one unit and inner nodes with biased radiance caches (the value in the node denotes the bias). Left subtrees are sampled with 90% probability. Right: MSE of path tracer sampling the tree, depending on actual sample count for different sample targets (dots represents the sample target). The dashed lines represent the optimal termination, which the algorithm does not attain in general.

factor (line 14) that allows the current MSE to be slightly larger. Thus, the prefix does not cancel entirely and is used to gauge the relative impact on the image. Additionally, in the limit (for infinite sample counts), termination will always be performed if the deferred bias is larger than the local bias. We clamp the deferred bias to 90% below the local bias (line 11, also accounting for the margin factor) to ensure that for large sample counts, termination is always deferred. Note that these terms only heuristically solve some of the issues arising from our approximation. More principled approaches are discussed in section 7.

Validation. Figure 5 shows a discrete test case of the MSE minimization. Compared to fig. 4, caches of inner nodes are now biased. The plot shows the MSE depending on the sample count for different sample targets (colored lines). The dot of each line represents the configured sample target, i.e. for that target it should have the lowest MSE among the other lines. We additionally plot the MSE for optimal termination decisions based on an exhaustive search (dashed lines). As can be seen, decisions are not optimal in general, since the optimal termination is often not found and the switch to another termination decision with more variance is made too early (discussed further in section 7).

4.4 Estimating local variance & bias

Our termination strategies depend on additional local statistics that are maintained for the individual cache records of the radiance cache structure. We will initially present how to estimate the local variance and later separate this quantity from the introduced bias due to averaging of contributions in a cache record.

Estimating local variance. To reiterate, the reformulated variance (eq. 1) consists of a sum of prefixed local variances that can be factored into a path-dependent prefix and a local variance $VE[g_j^{\text{suf}} \mid x_{j+1}]$. This separation allows to track the local variance individually for each partition, while the path-dependent prefix can be incorporated in an ad-hoc fashion. However, the nested expectation complicates direct estimation, especially for the second moment.

We leverage that the radiance cache itself can be used to approximate this expectation. Instead of collecting the moments of the suffix estimator, we construct an estimator that directly terminates at the next vertex and reads the stored radiance cache value. This leads to an iterative update scheme

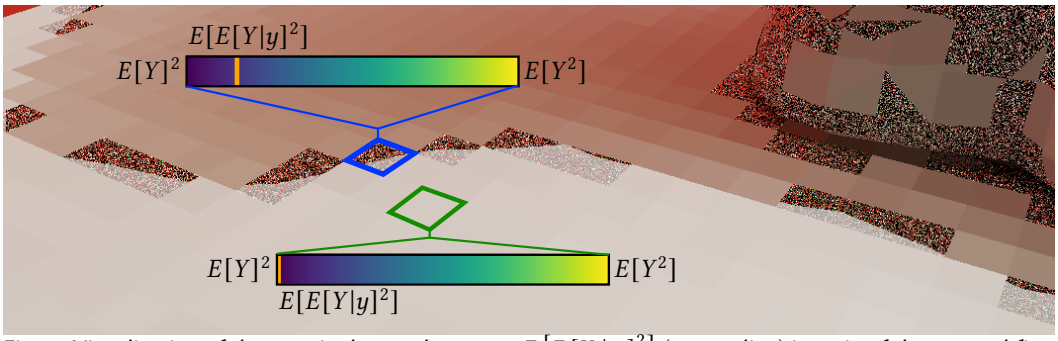


Fig. 6. Visualization of the marginal second moment $E\left[E\left[Y \mid y\right]^2\right]$ (orange line) in ratio of the squared first moment $E[Y]^2$ and the second moment $E\left[Y^2\right]$ for two different cache records. The green cell is uniformly lit, causing the marginal second moment to coincide with the squared first moment, indicating not much bias. The blue cell is crossed by a shadow edge, and the differently-lit regions cause the marginal second moment to shift more towards the second moment, indicating bias.

comparable to recent works [Deng et al. 2021; Pantaleoni 2020], where radiance cache values of previous frames are propagated forward (section 5).

Differentiating between variance and bias. As a cache record is shared between vertex pairs in a partition, its radiance estimate is generally biased for all of them. The plain variance estimate based on the first and second moment consistently overestimates the true variance in such a case. This is precisely due to the bias induced by the shared accumulation.

To separate both quantities, we explicitly compute the average variance and squared bias (the plain average bias is always zero) in a partition as the expectation of variance and bias over all vertex pairs. Generally, the average variance of a random variable Y conditioned to y ($Y = E[g_1^{\text{suf}} \mid x_0, x_1]$ and $y = (x_0, x_1)$ in our case) can be decomposed into the following moments (derivation in appendix A):

$$E\left[V\left[Y \mid y\right]\right] = E\left[Y^2\right] - E\left[E\left[Y \mid y\right]^2\right].$$

Similarly, the average squared bias of a random variable Y conditioned to y with respect to the expectation of Y can be decomposed into these moments:

$$E\left[\text{Bias}(Y \mid y, E[Y])^2\right] = E\left[E\left[Y \mid y\right]^2\right] - E\left[Y\right]^2.$$

Therefore, to distinguish between variance and bias, we need to track three moments: $E[Y]$, $E\left[Y^2\right]$ and $E\left[E\left[Y \mid y\right]^2\right]$. The first two are the usual first and second moments for variance or error computation. The latter is a marginal second moment. This term is what distinguishes between variance and bias, given that it is part of both the average variance and bias. Importantly, when adding both expressions together, the marginal second moment cancels, and we are left with only the first and second moment, which gives the original overestimated variance. Figure 6 shows a visualization of the three moments in the bunny scene. The region marked by the green square is uniformly lit. Here, the marginal second moment almost coincides with the squared first moment, thus all error is attributed to variance. In the region marked by the blue square, the marginal second moment shifts towards the second moment, thus some error is also attributed to bias. Note that in practice, the squared bias can often be much smaller than the variance. But since it does not shrink with sample count, it can still dwarf variance of the N-sample estimator at moderate sample counts.

Estimating marginal second moment. The marginal second moment is constructed as an expectation over a squared expectation $E\left[E\left[Y \mid y\right]^2\right]$. This makes it challenging to directly estimate it. A

naïve approach is to collect first moment estimates at a higher resolution with respect to the cache record. This allows us to quantify bias through the variation of the high-resolution first moment estimates, although this approach is both computation and memory intensive.

One approach to unbiasedly estimate this quantity without this memory overhead is to leverage that the expectation of the product of two independent realizations of a random variable is equal to the product of their expectations: $E[X_0X_1] = E[X_0]E[X_1]$. That is, we need to draw two independent samples from Y conditioned to y to obtain an unbiased estimate of $E\left[Y \mid y\right]^2$, the product is an unbiased estimate of squared expectation. This estimator tends to have very high variance, thus needing many samples. The application of U-statistics [Lee 1990], which have also been applied before in computer graphics [Kettunen et al. 2021], allows to amortize the cost of taking many independent samples by computing the product between all sample pairs, effectively resulting in a quadratic sample count.

Another approach is to take many independent samples conditioned to one randomly chosen y upfront and estimate the inner expectation. This only approximates the inner expectation due to a finite sample count, which results in a consistent overestimation of the marginal second moment (and as a consequence over-estimated variance and under-estimated bias) depending on the sample count.

In either case, estimating this quantity requires taking at least two (but ideally more) independent samples, complicating direct estimation in a forward path tracer, but the storage overhead of collecting moments at higher resolution is mitigated. We therefore run a separate compute dispatch to update cache records (see section 5 for implementation details).

5 Implementation details

We implemented our termination strategies in the Falcor rendering framework [Kallweit et al. 2022] in a basic path tracer with next event estimation (NEE) and BSDF-sampling, combined with MIS for direct illumination [Veach 1998]. The path tracer is implemented in a compute shader using ray queries. For the radiance cache, we use hashed grids [Binder et al. 2019; Hachisuka and Jensen 2010] as a spatial data structure.

Updating cache records. Typical radiance caching implementations focus on updating cache records actually accessed by the path tracer (e. g. by sparsely generating suffix paths when termination is performed at a cache record [Müller et al. 2021]). In our instance, however, local variance and bias estimation require relatively stable radiance estimates at the next path vertex. We therefore update cache records in a separate compute shader to ensure that cache record updates are independent of how often the path tracer reaches a cache record. The same importance sampling methods as in the path tracer are used to ensure that variance estimates match the path tracer. The radiance cache maintains four vertices per cache record to be used by the compute dispatch to update the radiance estimate of the cache record, although other storage layouts independent of the radiance cache structure are also thinkable. The path tracer stores primary vertices with a 1% probability, while the compute dispatch itself also stores hit vertices in order to explore the scene and frequently refresh the set of representative vertices.

The update scheme is iterative: At each vertex location, a BSDF sample is drawn, and the incoming radiance is estimated by evaluating the radiance cache at the hit location, thus propagating radiance over multiple frames. This approach is comparable to recent works [Deng et al. 2021; Pantaleoni 2020]. We also exploit this iterative update scheme to compute the local variance (section 4.4), since the contained expectation over the path suffix is conveniently approximated with the radiance cache.

Computation of the marginal second moment requires drawing multiple samples at the same vertex location and computing products between all sample pairs. This is straightforward with a separate compute dispatch because we can leverage SIMT-layout of the hardware and draw 32 samples for a given vertex in one subgroup and use register permutations to exchange sample contributions, thus reducing register pressure and increasing coherence.

Relative variance bounding. Variance is a scale-dependent quantity, which complicates finding good thresholds. If some estimate of pixel brightness is available (e.g. through denoising or the radiance cache itself), the variance threshold can be interpreted as a relative quantity and the absolute threshold that is needed by the algorithm in fig. 3 can be recovered by multiplying with the squared pixel brightness. In our implementation, we use the radiance cache value of the first hit to approximate pixel brightness.

Interpolating cache records. Directly using the radiance estimate of a single cache record corresponding to a vertex leads to noticeable grid artifacts (fig. 1 right). These artifacts also tend to be preserved by denoisers. A straightforward and inexpensive solution is to perform stochastic interpolation between neighboring cache records by jittering the vertex position before access [Binder et al. 2019]. This approach introduces additional variance that our variance estimates do not account for. However, since estimates of neighboring cells are usually similar in scale, this variance can likely be neglected when comparing to the variance of path tracing.

6 Evaluation

We initially analyze the behavior of our termination strategies (with parameters t for the variance threshold and s for the MSE sample target) and then continue with a comparative evaluation to the area-spread heuristic [Müller et al. 2021] (relative spread of $c = 0.01$, unless stated otherwise) with respect to quality and performance. We use both custom and openly available scenes [Bitterli 2016] and modified them if needed to force more complex light paths that require more indirect bounces, as those are the main use case for our techniques. We chose to only use diffuse materials for now. While the radiance cache can be trained for specular data, e.g., using a directional histogram, it becomes very memory-intensive. Performance measurements were taken on an Nvidia RTX 3070 Ti at a resolution of 1920×1080. As we focus on static scene configurations, we fill and propagate the radiance cache in a preprocess until it reaches an equilibrium state.

Variance bounding. Similar to our detached test case (fig. 4) we evaluate behavior of the variance bounding strategy in the Veach Door scene (fig. 7). We measure variance and bias depending on the variance threshold (fig. 7, left) as well as path length (fig. 7, right). Note that path length accounts for variance interpolation in this case. To analyze termination behavior depending on different importance sampling strategies, we introduce probabilistic path termination through Russian-Roulette [Arvo and Kirk 1990] with different survival probabilities (100%, 60% and 20%). We can generally observe a similar behavior as in the detached test case: For all survival probabilities, the variance bound is initially closely followed and eventually detaches because the underlying estimator has finite variance. Path length increases while bias decreases for increasing thresholds. Of note is that bias is dependent on the survival probability: Since low survival probabilities introduce more variance, paths are terminated earlier to maintain a given variance thresholds, increasing variance.

The controlled bias-variance tradeoff in our technique can be used to generate images that are more suitable for denoising. We use the SVGF [Schied et al. 2017] denoiser and use its result after four still frames. The output of the denoiser for different relative variance thresholds with stochastic interpolation are shown in fig. 8. A low threshold, e.g., taking the biased estimate from the radiance

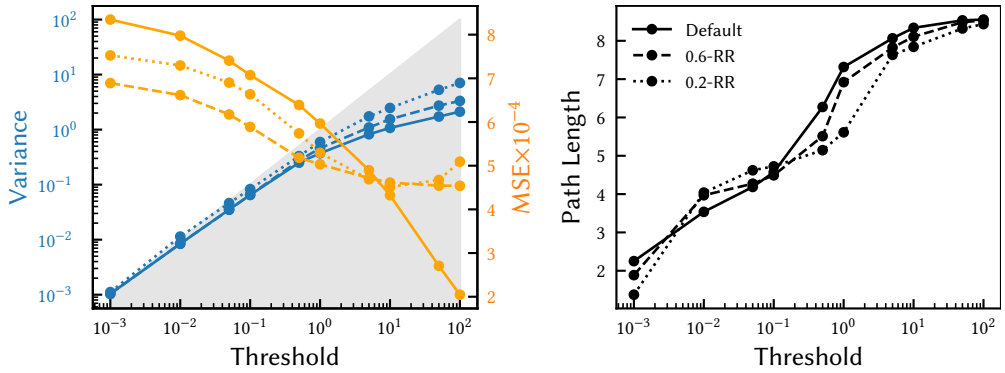


Fig. 7. Left: Variance bounding on the Veach Door scene for different Russian-Roulette survival probabilities. A lower variance threshold (blue) increases the MSE (orange) due to biased radiance estimates. Note that the MSE increases for low survival probabilities at high thresholds due to images not being completely converged. Right: Increased variance thresholds increase path length.

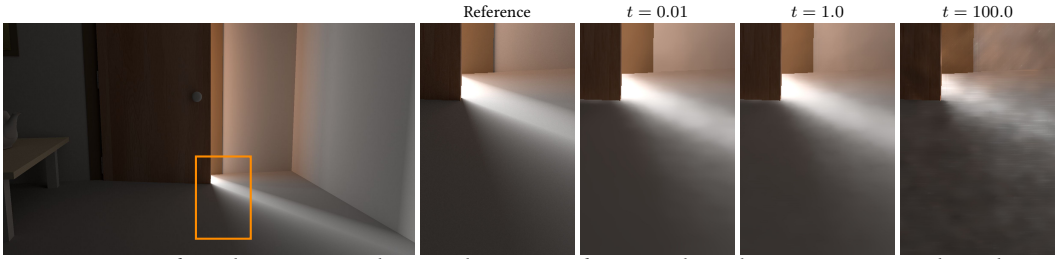


Fig. 8. Testcase of Veach Door scene showing the impact of variance bounding termination with stochastic interpolation on denoising. Higher thresholds increase variance, leading to more unstable output.

cache at the primary hit, produces structured artifacts. A high threshold, e.g., the noisy output of the plain path tracer, produces temporally unstable images. Choosing a suitable variance threshold achieves a balance that hides cache artifacts but also provides a more stable input image for the denoiser.

MSE minimization. We evaluate how the MSE minimization strategy behaves in the Bunny and Classroom scenes in fig. 9, similar to the detached test case in fig. 5. The lines represent individual sample targets (signified by the marker), plotted as MSE over sample count. Ideally, each line should be globally minimal at its sample target, which can be partially observed. Especially the Bunny scene contains overshoots, where path terminations are made too late and results in overall higher error. However, for sample counts below 100, termination decisions are generally favorable. This is likely because around this region, deferring happens at primary vertices, where the approximation is more accurate.

Comparison to area-spread heuristic. A comparison for different scenes between our techniques and the area-spread heuristic [Müller et al. 2021] is shown in fig. 10. All comparisons are with equal sample count and using stochastic interpolation. The main difference between our techniques and the area-spread heuristic is that we explicitly estimate error to optimize termination, while the area-spread heuristic estimates how well unquantified error is hidden through an approximate ray differential. The area-spread heuristic never terminates at a primary hit, which makes it less prone to discretization artifacts, but after that it generally terminates early, except for corner

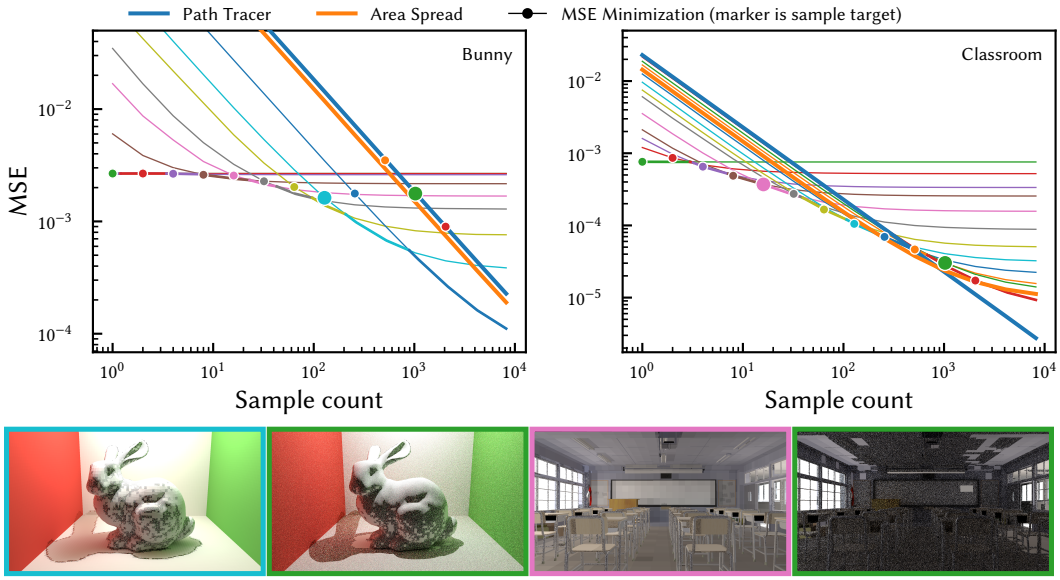


Fig. 9. Behavior of MSE minimization in two scenes (Bunny, Classroom) at different sample targets, plotted as MSE depending on sample count. Each line represents one sample target, signified by the marker. The plain path tracer and area-spread heuristics are also given for reference. For the enlarged markers, one-sample images are shown below.

regions where distances to the next vertex can be very short. Earlier termination can sometimes be beneficial, as scenes like corridor show.

This aspect is even more pronounced in the Veach door scene (fig. 11), where the incoming lighting from the door gap is estimated with high variance, rendering the output image unusable for denoising. The variance bounding strategy has explicit knowledge of the local variance, and can therefore terminate directly at the primary vertex. On the other hand, the region under the door is better handled by the area-spread heuristic, since variance bounding estimates high variance in this region, leading to earlier termination.

Figure 12 summarizes the prime cases where termination at the secondary vertex is not beneficial: (1) when terminating at the secondary vertex causes high variance (left image). In this instance, it is caused by incoming radiance at the surface being confined to a small solid angle, which is challenging to estimate with plain BSDF sampling. (2) When terminating at secondary or later vertices introduces high bias (right image). In this instance, a glossy reflection at the secondary vertex introduces bias. No parameterization of area-spread can give satisfactory results for both cases, while the output of variance bounding gets closer to the reference.

Performance measurements. Table 1 shows performance measurements of our variance bounding and MSE minimization strategies compared to the area-spread heuristic with different parameterizations. We report execution times of the path tracer and propagation pass separately, since updating the cache is mostly an orthogonal issue. We additionally report the average path length depending on the parameterization. We generally observe the area-spread heuristic (for its recommended parameter $c = 0.01$) to terminate paths mostly at the first indirection, which is consistent with an average path length of almost 2 in both scenes. Larger variance thresholds generally introduce longer paths, resulting in longer execution times. Since our methods need to access the radiance cache from global memory at every surface interaction to decide whether to terminate, we have an

Fig. 10. Equal-sample comparison of termination strategies with and without denoising for different scenes: The plain path tracer vs. our variance bounding (relative variance threshold $t = 1.0$) vs. our MSE minimization (sample target $s = 100$) vs. the area-spread heuristic ($c = 0.01$). The first row shows the 1spp output, the second row the output of the denoiser [Schied et al. 2017] after four still frames.

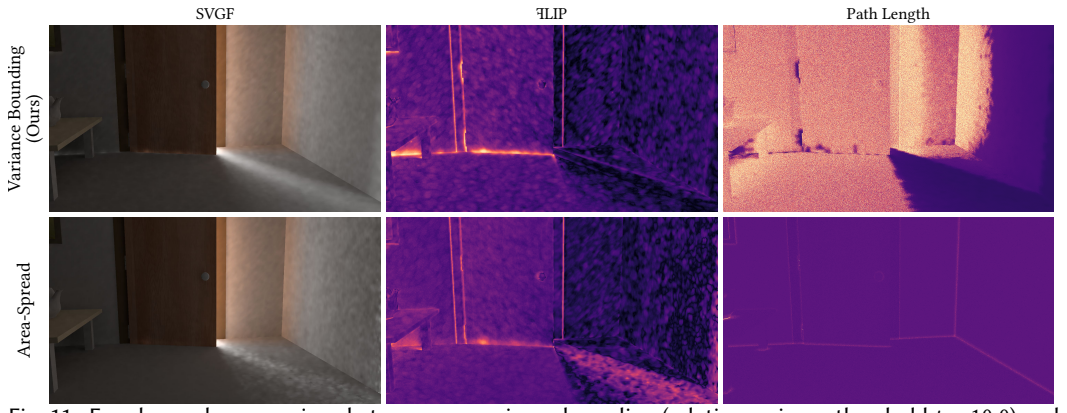


Fig. 11. Equal-sample comparison between our variance bounding (relative variance threshold $t = 10.0$) and the area-spread termination. Left: output of the denoiser. Middle: ꟻLIP error metric [Andersson et al. 2021] compared to the reference. Right: visualization of the average path length per pixel. Apart from the edges of the room, the area-spread heuristic terminates quite early. Our variance bounding can adapt to the variance caused by indirect illumination.

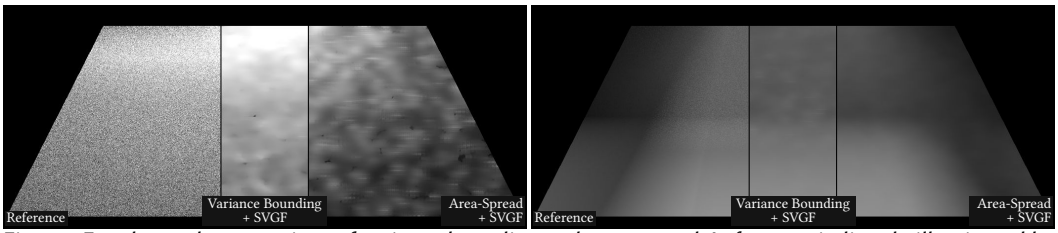


Fig. 12. Equal-sample comparison of variance bounding and area-spread. Left: scene indirectly illuminated by a small gap in a box containing a light source, causing high variance when not terminating at primary vertex. Right: surface indirectly illuminated by glossy reflection of an area light, causing bias when terminating at secondary vertex. Area-spread ($c = 0.01$) terminates at the second vertex in both cases, causing high variance and bias, respectively. Variance bounding ($t = 1.0$) terminates at the primary vertex in the left case and at the light source in the right case, resulting in lower variance and bias, respectively.

Table 1. Frame time comparison between our variance bounding (absolute variance threshold t), our MSE minimization and the area-spread heuristic. The threshold indirectly also bounds the path length. Our methods share the same radiance propagation algorithm. The area-spread heuristic does not require the additional statistics, resulting in lower execution times of the propagation pass.

Scene	Method	Total Frame [ms]	Tracer [ms]	Propagation [ms]	Path Length
Door	Var. Bound. ($t = 0.01$)	21.71	6.05	10.68	4.05
	Var. Bound. ($t = 0.10$)	22.64	6.94		4.86
	Var. Bound. ($t = 1.00$)	26.08	10.46		7.41
	MSE Min. ($s = 100$)	16.90	0.67		1.01
	Area-Spread ($c = 0.01$)	17.85	1.86	9.86	1.97
	Area-Spread ($c = 100$)	19.85	4.73		3.18
Corridor	Var. Bound. ($t = 0.01$)	12.46	1.39	6.22	1.97
	Var. Bound. ($t = 0.10$)	15.42	4.35		4.24
	Var. Bound. ($t = 1.00$)	15.70	4.63		4.88
	MSE Min. ($s = 100$)	13.23	0.53		1.00
	Area-Spread ($c = 0.01$)	11.96	1.48	5.64	1.98
	Area-Spread ($c = 100$)	15.02	4.49		4.18

additional runtime overhead compared to the area-spread heuristic. The overhead seems to be small in practice, but more pronounced with more indirections, given that cache access at the primary vertex is likely faster due to memory coherence. The propagation pass in the case of area-spread has lower execution times, since it only needs to maintain radiance estimates, as opposed to additional variance and bias estimates. An additional speedup can be expected from lowering the sample count, since plain radiance estimates have lower variance.

7 Discussion & Future Work

Path termination. The variance bounding termination strategy maximizes path length under a variance constraint. The rationale is that longer paths lead to lower bias, since errors due to radiance caching are reduced at deeper path vertices. However, since the algorithm is oblivious to the actual error reduction, there is no guarantee that error is reduced significantly. We have shown instances where this approach is indeed meaningful: Forcing early termination when local variance is very large (fig. 11) and continuing traversal to avoid local bias if the variance threshold permits (figs. 11 and 12).

Optimality of MSE minimization. As we only track the second moment of bias as an upper bound of the squared first moment, the impact of bias is overestimated, leading to path termination being deferred further than necessary (this is already observable in the theoretical test case). This

approximation foregoes the non-local cancelation of biases in unrelated parts of the path tree, since all biases are positivized through squaring. This especially affects gradient biases that are likely hidden well through cache interpolation. Finding better approximations to capture these interactions can lead to more informed termination decisions.

Efficient bias estimation. While the estimator of the marginal second moment eliminates the prohibitive memory cost of a high-resolution intermediate representation, it possesses quite strong variance in general, requiring large sample counts for stable estimation. While the application of U-statistics offsets this cost to some degree, it is still not applicable in a dynamic context where short sample histories have to be maintained in order to react to changes in illumination. More approximate approaches could be conceivable, as well as additional filtering on the raw moment estimates to stabilize the output at lower sample counts.

Efficiency awareness. Minimizing MSE has a similar issue as variance bounding in that paths are continued even if the error reduction is marginal. While we heuristically approach this problem with the margin factor (terminating even if the MSE when continuing would be slightly smaller), a more principled approach to optimize efficiency directly could prove fruitful.

Applications of bias estimation. Knowing biases in radiance caching can be a useful tool for other purposes, too. One example is to control refinement of the cache structure towards regions of high bias. This might also allow for more efficient estimation, given that less cache records compared to a naïve grid would be present.

8 Conclusion

Optimizing path termination into radiance caches has the potential to give dramatic improvements in image quality for a given sample budget and reduce overall rendering times. We have conceived a mathematical framework to tackle the bias-variance tradeoff using local termination decisions and presented two algorithms, that optimize the tradeoff differently. The variance bounding termination strategy is capable of maintaining a given variance target with implicit bias reduction. With the MSE minimization strategy, we have made an initial step towards exploiting the cache structure to achieve lower error for a given number of samples to accumulate, which can be especially useful for highly limited sample budgets. While the results are promising, future work should focus on finding tighter approximations to achieve better tradeoffs.

Acknowledgments

We thank the anonymous reviewers for their guidance and feedback and Vincent Schüßler for numerous insightful discussions.

References

Pontus Andersson, Jim Nilsson, Peter Shirley, and Tomas Akenine-Möller. 2021. Visualizing Errors in Rendered High Dynamic Range Images. *Eurographics 2021 - Short Papers* (2021). https://doi.org/10.2312/EGS.20211015

James Arvo and David Kirk. 1990. Particle transport and image synthesis. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH90)*. ACM. https://doi.org/10.1145/97879.97886

Philippe Bekaert, Philipp Slusallek, Ronald Cools, Vlastimil Havran, and Hans-Peter Seidel. 2003. *A custom designed density estimatorfor light transport.* Technical Report.

Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2019. Massively Parallel Path Space Filtering. https://doi.org/10.48550/ARXIV.1902.05942

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

Mark R. Bolin and Gary W. Meyer. 1997. *An Error Metric for Monte Carlo Ray Tracing.* Springer Vienna, 57–68. https://doi.org/10.1007/978-3-7091-6858-5_6

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics* 36, 4 (July 2017), 1–12. https://doi.org/10.1145/3072959.3073601

Xi Deng, Miloš Hašan, Nathan Carr, Zexiang Xu, and Steve Marschner. 2021. Path graphs. *ACM Transactions on Graphics* 40, 6 (dec 2021), 1–15. https://doi.org/10.1145/3478513.3480547

Marc Droske, Johannes Hanika, Jiří Vorba, Andrea Weidlich, and Manuele Sabbadin. 2023. Path tracing in Production: The Path of Water. In *ACM SIGGRAPH 2023 Courses (SIGGRAPH '23)*. ACM. https://doi.org/10.1145/3587423.3595519

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Transactions on Graphics* 27, 3 (Aug. 2008), 1–10. https://doi.org/10.1145/1360612.1360632

Toshiya Hachisuka and Henrik Wann Jensen. 2010. Parallel progressive photon mapping on GPUs. In *ACM SIGGRAPH ASIA 2010 Sketches (SA '10)*. ACM. https://doi.org/10.1145/1899950.1900004

Saeed Hadadan, Shuhong Chen, and Matthias Zwicker. 2021. Neural radiosity. *ACM Transactions on Graphics* 40, 6 (Dec. 2021), 1–11. https://doi.org/10.1145/3478513.3480569

Henrik Halen, Andreas Brinck, Kyle Hayward, and Xiangshun Bei. 2021. Global Illumination Based on Surfels. Presentation.

Henrik Wann Jensen. 1996. *Global Illumination using Photon Maps*. Springer Vienna, 21–30. https://doi.org/10.1007/978-3-7091-7484-5_3

J. L. W. V. Jensen. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica* 30, 0 (1906), 175–193. https://doi.org/10.1007/bf02418571

James T. Kajiya. 1986. The rendering equation. *ACM SIGGRAPH Computer Graphics* 20, 4 (Aug. 1986), 143–150. https://doi.org/10.1145/15886.15902

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

Anton S. Kaplanyan and Carsten Dachsbacher. 2013. Adaptive progressive photon mapping. *ACM Transactions on Graphics* 32, 2 (April 2013), 1–13. https://doi.org/10.1145/2451236.2451242

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2016. *Path Space Filtering*. Springer International Publishing, 423–436. https://doi.org/10.1007/978-3-319-33507-0_21

Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. 2021. An unbiased ray-marching transmittance estimator. *ACM Transactions on Graphics* 40, 4 (July 2021), 1–20. https://doi.org/10.1145/3450626.3459937

J. Krivanek, P. Gautron, S. Pattanaik, and K. Bouatouch. 2005. Radiance Caching for Efficient Global Illumination Computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (Sept. 2005), 550–561. https://doi.org/10.1109/tvcg.2005.83

A. J. Lee. 1990. *U-statistics: Theory and Practice*. M. Dekker. 302 pages.

Julio Marco, Adrian Jarabo, Wojciech Jarosz, and Diego Gutierrez. 2018. Second-Order Occlusion-Aware Volumetric Radiance Caching. *ACM Transactions on Graphics* 37, 2 (April 2018), 1–14. https://doi.org/10.1145/3185225

Don P. Mitchell. 1987. Generating antialiased images at low sampling densities. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87)*. ACM. https://doi.org/10.1145/37401.37410

Jacob Munkberg, Jon Hasselgren, Petrik Clarberg, Magnus Andersson, and Tomas Akenine-Möller. 2016. Texture space caching and reconstruction for ray tracing. *ACM Transactions on Graphics* 35, 6 (Nov. 2016), 1–13. https://doi.org/10.1145/2980179.2982407

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–15. https://doi.org/10.1145/3528223.3530127

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics* 40, 4 (July 2021), 1–16. https://doi.org/10.1145/3450626.3459812

Jacopo Pantaleoni. 2020. Online Path Sampling Control with Progressive Spatio-temporal Filtering. *SN Computer Science* 1, 5 (Aug. 2020). https://doi.org/10.1007/s42979-020-00291-z

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-aware path guiding. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392441

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Philippe Weier, and Philipp Slusallek. 2022. EARS: efficiency-aware russian roulette and splitting. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–14. https://doi.org/10.1145/3528223.3530168

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics (HPG '17)*. ACM. https://doi.org/10.1145/3105762.3105770

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-based error control for irradiance caching. *ACM Transactions on Graphics* 31, 6 (Nov. 2012), 1–10. https://doi.org/10.1145/2366145.2366212

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 3 (July 2022), 1–22. https://doi.org/10.1145/3543870

Eric Veach. 1998. *Robust monte carlo methods for light transport simulation.* Ph. D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.

Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. 1988. A ray tracing solution for diffuse interreflection. *ACM SIGGRAPH Computer Graphics* 22, 4 (June 1988), 85–92. https://doi.org/10.1145/378456.378490

Daniel Wright, Krzysztof Narkowicz, and Patrick Kelly. 2022. Real-time Global Illumination in Unreal Engine 5. Presentation.

A Moment derivation for average variance & bias

Here, we derive the moments of average variance and squared bias of a random variable Y conditioned to y (section 4.4). The squared bias is defined with respect to $E[Y]$:

$$\begin{aligned} E\left[V\left[Y \mid y\right]\right] &= E\left[V\left[Y \mid y\right]\right] \\ &= E\left[E\left[Y^2 \mid y\right] - E\left[Y \mid y\right]^2\right] \\ &= E\left[E\left[Y^2 \mid y\right]\right] - E\left[E\left[Y \mid y\right]^2\right] \\ &= E\left[Y^2\right] - E\left[E\left[Y \mid y\right]^2\right], \end{aligned}$$

$$\begin{aligned} E\left[\mathrm{Bias}(Y \mid y, E[Y])^2\right] &= E\left[\left(E\left[Y \mid y\right] - E\left[Y\right]\right)^2\right] \\ &= E\left[E\left[Y \mid y\right]^2 - 2E\left[Y \mid y\right]E\left[Y\right] + E\left[Y\right]^2\right] \\ &= E\left[E\left[Y \mid y\right]^2\right] - E\left[2E\left[Y \mid y\right]E\left[Y\right]\right] + E\left[E\left[Y\right]^2\right] \\ &= E\left[E\left[Y \mid y\right]^2\right] - 2E\left[Y\right]^2 + E\left[Y\right]^2 \\ &= E\left[E\left[Y \mid y\right]^2\right] - E\left[Y\right]^2. \end{aligned}$$