Multiple Importance Reweighting for Path Guiding

ZHIMIN FAN, State Key Lab for Novel Software Technology, Nanjing University, China
YIMING WANG, State Key Lab for Novel Software Technology, Nanjing University, China
CHENXI ZHOU, State Key Lab for Novel Software Technology, Nanjing University, China
LING-QI YAN, University of California, Santa Barbara, United States of America
YANWEN GUO, State Key Lab for Novel Software Technology, Nanjing University, China
JIE GUO*, State Key Lab for Novel Software Technology, Nanjing University, China

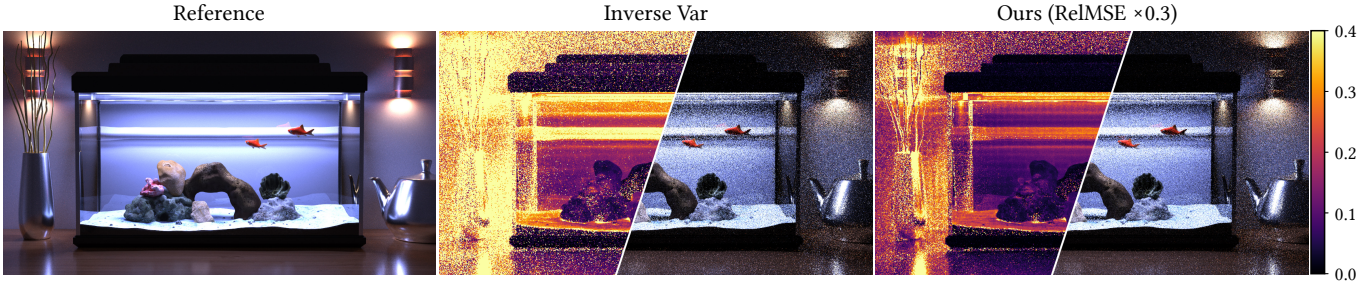

Fig. 1. Rendering an aquarium scene (24 spp) featuring complex indirect illumination using Practical Path Guiding [Müller et al. 2017]. The samples produced during each iteration are merged into the final image through inverse-variance-weighted combination [Müller 2019] and our proposed path-level *multiple importance reweighting*, respectively. The variance maps show clear improvements in variance reduction of our method compared to the baseline.

Contemporary path guiding employs an iterative training scheme to fit radiance distributions. However, existing methods combine the estimates generated in each iteration merely within image space, overlooking differences in the convergence of distribution fitting over individual light paths.

This paper formulates the estimation combination task as a path reweighting process. To compute spatio-directional varying combination weights, we propose *multiple importance reweighting*, leveraging the importance distributions from multiple guiding iterations. We demonstrate that our proposed path-level reweighting makes guiding algorithms less sensitive to noise and overfitting in distributions. This facilitates a finer subdivision of samples both spatially and temporally (i.e., over iterations), which leads to additional improvements in the accuracy of distributions and samples.

Inspired by adaptive multiple importance sampling (AMIS), we introduce a simple yet effective mixture-based weighting scheme with theoretically guaranteed consistency, demonstrating good practical performance compared to alternative weighting schemes. To further foster usage with high sample rates, we introduce a hyperparameter that controls the size of sample storage. When this size limit is exceeded, low-valued samples are splatted during rendering and reweighted using a partial mixture of distributions. We found limiting the storage size reduces memory overhead and keeps variance reduction and bias comparable to the unlimited ones.

Our method is largely agnostic to the underlying guiding method and compatible with conventional pixel reweighting techniques. Extensive evaluations underscore the feasibility of our approach in various scenes, achieving variance reduction with negligible bias over state-of-the-art solutions within equal sample rates and rendering time.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Path guiding

ACM Reference Format:
Zhimin Fan, Yiming Wang, Chenxi Zhou, Ling-Qi Yan, Yanwen Guo, and Jie Guo. 2025. Multiple Importance Reweighting for Path Guiding. *ACM Trans. Graph.* 44, 4, Article 1 (August 2025), 11 pages. https://doi.org/10.1145/3618360

1 INTRODUCTION

Monte Carlo path tracing [Fascione et al. 2018; Keller et al. 2015] solves the rendering equation [Kajiya 1986] via stochastic sampling, which is notoriously time-consuming and prone to noise when applied to complex scenes. To accelerate convergence, path guiding [Herholz et al. 2016, 2019; Huang et al. 2024; Müller et al. 2019; Rath et al. 2020; Reibold et al. 2018; Ruppert et al. 2020; Schüßler et al. 2022] employs an iterative training process, progressively refining the proposal distribution by leveraging information from historical samples, which could yield substantial noise reduction. Consequently, guided path tracers robustly handle a wide variety of light transport effects [Fan et al. 2023, 2024; Rath et al. 2023], play a pivotal role in film production [Vorba et al. 2019], and extend their impact even to real-time applications [Derevyannykh 2022].

In early guiding algorithms, the estimates generated during training were either discarded [Ruppert et al. 2020; Vorba et al.

*Corresponding author.

Authors' addresses: Zhimin Fan, zhiminfan2002@gmail.com, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Yiming Wang, 02yimingwang@gmail.com, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Chenxi Zhou, 502023320017@smail.nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Ling-Qi Yan, lingqi@cs.ucsb.edu, University of California, Santa Barbara, Santa Barbara, United States of America; Yanwen Guo, ywguo@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Jie Guo, guojie@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China.

2014] or assigned per-iteration constant weights when producing the final rendering results [Reibold et al. 2018]. The optimal per-pixel weights are proportional to the reciprocal pixel variance. Since population variance is not accessible, sample variance is alternatively used to compute these weights and often averaged over the whole image to reduce bias [Müller 2019]. However, given the adaptive nature of proposal distributions, the change of variance across iterations varies across path space, as showcased in Fig. 2, which suggests that the optimal weight should also fluctuate for different paths. Even within a pixel exhibiting high variance, certain regions of its support in path space may still be sufficiently sampled, leading to a well-estimated integral within that specific region.

This paper presents a new framework for combining all samples generated during training and rendering through a path-level reweighting procedure. We introduce *multiple importance reweighting*, which computes the weight for each sample as if they were drawn from various techniques in multiple importance sampling (MIS) [Grittmann et al. 2022; Kondapaneni et al. 2019; Veach and Guibas 1995b]. These weights are designed to capture the relative efficacy of each iteration for a particular path, leveraging guiding distributions and auxiliary information such as sample rates and overall variance [Grittmann et al. 2019].

Unlike multiple importance sampling where techniques are usually conceptually distinct [Grittmann et al. 2021; Kondapaneni et al. 2019; Veach and Guibas 1995b; West et al. 2020], guiding distributions all fit the same target distribution. However, these fitted distributions often suffer from noise and may fail to capture some patterns of targets in certain iterations. Our proposed reweighting scheme effectively serves as mixing several noisy distributions into a more robust one when computing importance weights, as if they were drawn from the mixture distribution. This could result in more stable importance weights, making guiding algorithms less susceptible to overfitting in certain iterations. Consequently, a finer spatio-directional subdivision and sample allocation can be safely employed even without incremental learning methods [Ruppert et al. 2020]. Additionally, we introduce a simple yet effective mixture-based weighting approach inspired by adaptive multiple importance sampling (AMIS) [Cornuet et al. 2009; El-Laham et al. 2019; Marin et al. 2019; Tokuyoshi et al. 2010] that combines samples from adaptive importance samplers [Cappé et al. 2008; Elvira and Martino 2021], which enjoys guaranteed consistent convergence. Our evaluation indicates that the proposed path-level reweighting could lead to variance reduction with minimal computational overhead, especially for short renderings. The main drawback of our method is that it requires storing path samples and historical distributions. To alleviate this issue, we further explore how to reduce the memory overhead using constant sample storage controlled by user parameters, whose performance is identical to storing all samples in our tests. This soundly extends the range of scenarios that path guiding can robustly handle. Nevertheless, due to the computational overhead, we found the benefit is visible primarily for low sample rates.

In summary, our contribution includes:

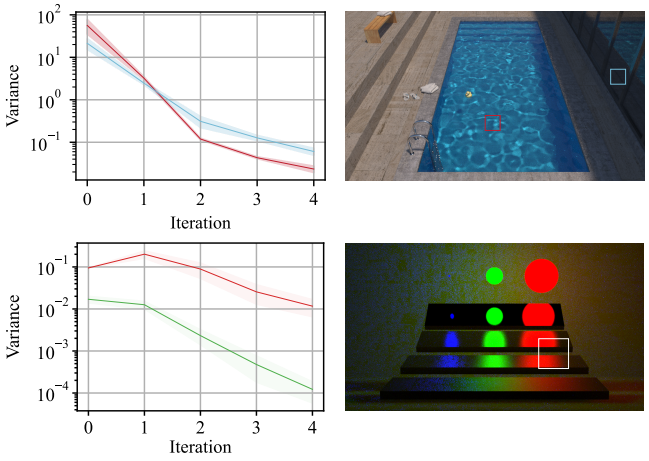

Fig. 2. Variance changes across iterations differ across different regions due to the evolution of distributions. We show variance averaged over 6 independent runs and the standard derivation of variance. The bottom example presents variance from the red and the green emitters, respectively.

- Propose a path-level sample reweighting framework for path guiding, where the weights can be computed from guiding distributions of multiple iterations.
- Demonstrate that path reweighting stabilizes individual sample weights, making the estimator more robust to finer spatial and temporal subdivisions of training samples.
- Develop an efficient mixture-based reweighting algorithm with provable consistency and reduced overhead.

2 MOTIVATION

Physically-based rendering computes the Monte Carlo estimation of the path integral [Veach 1997]

$$I = \int_{\mathcal{P}} f(\overline{\boldsymbol{x}}) \mathrm{d}\overline{\boldsymbol{x}}, \tag{1}$$

where $f(\overline{\boldsymbol{x}})$ is the measurement contribution function and $\mathcal{P}$ is path space comprised of all paths $\overline{\boldsymbol{x}}$. A typical path tracer guided by historical path samples [Müller et al. 2017; Ruppert et al. 2020; Vorba et al. 2014] uses a sequence of distributions $p_1(\overline{\boldsymbol{x}}), p_2(\overline{\boldsymbol{x}}), ..., p_M(\overline{\boldsymbol{x}})$ trained iteratively as proposals. Each proposal is typically a mixture of a fitted distribution and a defensive sampling technique (e.g., BSDF sampling). The importance weight of a sample drawn from p_i can be expressed as $f(\overline{\boldsymbol{x}})/p_i(\overline{\boldsymbol{x}})$, and the Monte Carlo estimation of Eq. (1) is often given using the samples from the last iteration, i.e.,

$$\left\langle \int_{\mathcal{P}} f(\overline{\boldsymbol{x}}) \mathrm{d}\overline{\boldsymbol{x}} \right\rangle = \frac{1}{n_M} \sum_{j=1}^{n_M} \frac{f(\overline{\boldsymbol{X}}_{M,j})}{p_M(\overline{\boldsymbol{X}}_{M,j})}. \tag{2}$$

Here, we drawn n_i independent samples from p_i and $\overline{\boldsymbol{X}}_{i,j}$ refers to the j-th sample in i-th iteration. The importance weight of these samples is computed only from the distribution of the last iteration.

Although many samples are generated during training (i.e., iteration $1 \sim M-1$), considering the potential high variance of

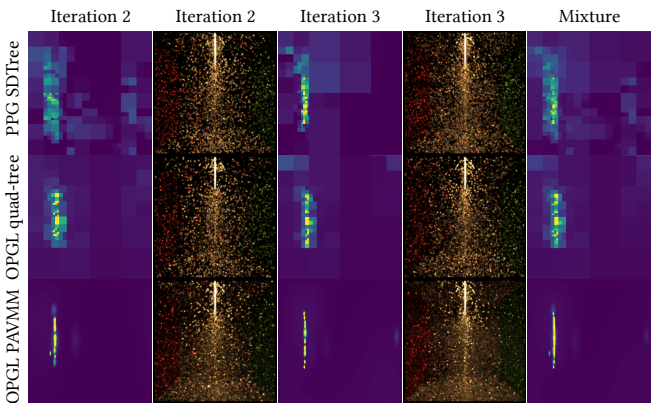

Fig. 3. Fitting the radiance distribution predominated by a line-shaped light source in a glass shade. We visualize distributions at the center of the floor. Fitted distributions, especially in *early* iterations, are extremely noisy, which can be alleviated by using a mixture across iterations. Three rows are fitted using the official implementation of PPG [Müller et al. 2017], the quadtree in the OpenPGL library [Herholz and Dittebrandt 2022], and the parallax-aware mixture distribution [Ruppert et al. 2020] in OpenPGL, respectively. The second and fourth columns showcase renderings using these distributions at 64 spp.

these estimations, they are not splatted into the image. Thus, the overhead of training remains a limitation of conventional guiding approaches, and the primary solution lies in amalgamating all these estimations produced during training into the final rendering image.

Previous methods [Müller 2019] combine the estimations given by each iteration in image space. This introduces a constant (over a pixel or the whole image) weighting factor w_i. The optimal choice of this (globally constant) weight is proportional to the reciprocal of pixel variance [Havran and Sbert 2014; Müller 2019].

However, since the proposal distribution changes across iterations, convergence rates typically vary across different parts of path space, as illustrated in Fig. 2. Notably, this issue cannot be completely addressed by employing per-pixel variance, as it only captures differences in image space and treats a pixel as a whole entity, regardless of the energy variation in path space. Even in a pixel with high variance, some parts of path space may still be well sampled, and samples could form a good estimation within that region. Nevertheless, all these samples will receive a low weight when employing per-pixel variance-based weighting. Moreover, per-pixel variance introduces significant bias due to strong correlations between weights and estimations. Additionally, image-space combinations neglect information regarding the proposal distribution, which could also provide information about the quality of samples in each iteration beyond the overall variance of estimations.

3 METHODOLOGY

The above insights prompt us to leverage the densities of proposal distributions to compute spatio-directional varying combination

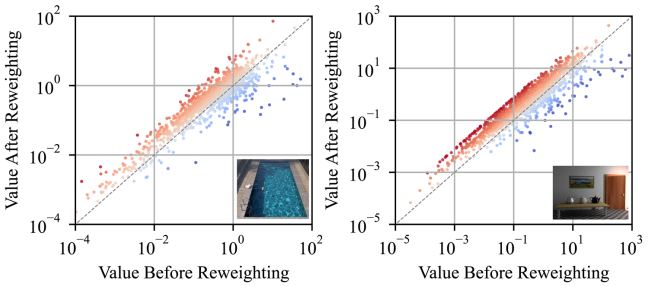

Fig. 4. Scatter plots of sample values (i.e., importance weights) before and after reweighting. Our reweighting procedure effectively decreases the weights (colored in blue) of many samples with high values, while increasing the weights (colored in red) of most samples with low values.

weights. To begin with, we present a formal definition of path-level sample reweighting for guiding approaches.

3.1 Path-level reweighting

Formally, we combine the path samples by introducing a path-level weighting factor $w_i(\overline{\boldsymbol{x}})$, which varies for different light paths over the whole path space. This leads to the following estimation:

$$\left\langle \int_{\mathcal{P}} f(\overline{\boldsymbol{x}}) \mathrm{d}\overline{\boldsymbol{x}} \right\rangle = \sum_{i=1}^{M} \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{w_i(\overline{\boldsymbol{X}}_{i,j}) f(\overline{\boldsymbol{X}}_{i,j})}{p_i(\overline{\boldsymbol{X}}_{i,j})}. \tag{3}$$

Since paths are incrementally constructed in guided path tracing, the probability density is a product of several conditional densities:

$$p_i(\overline{\boldsymbol{x}}) = p_i(\boldsymbol{x}_1) p_i(\boldsymbol{x}_2|\boldsymbol{x}_1) \prod_{j=2}^{N-1} p_i(\boldsymbol{x}_{j+1}|\boldsymbol{x}_j, \boldsymbol{x}_{j-1}). \tag{4}$$

Here, N represents the length of the path $\overline{\boldsymbol{x}}$ and $\boldsymbol{x}_i$ denotes the i-th vertex of $\overline{\boldsymbol{x}}$, with $\boldsymbol{x}_1$ being the lens vertex. Using the fact that the position of each vertex $\boldsymbol{x}_j$ is decided by vertex $\boldsymbol{x}_{j-1}$ and the direction $\boldsymbol{\omega}_{j-1}$, we can rewrite the above equation as

$$p_i(\overline{\boldsymbol{x}}) = p_i(\boldsymbol{x}_1) G(\boldsymbol{x}_1, \boldsymbol{x}_2) p_i(\boldsymbol{\omega}_1|\boldsymbol{x}_1) \prod_{j=2}^{N-1} G(\boldsymbol{x}_j, \boldsymbol{x}_{j+1}) p_i(\boldsymbol{\omega}_j|\boldsymbol{x}_j, \boldsymbol{x}_{j-1}), \tag{5}$$

where $\boldsymbol{\omega}_i$ represents the direction from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{i+1}$ and $G(\boldsymbol{x}_j, \boldsymbol{x}_{j+1})$ is the geometric factor [Veach and Guibas 1995a] including the mutual visibility term. The weighting factors should be normalized across iterations, i.e., for all $\overline{\boldsymbol{x}}$, $\sum_{i=1}^{M} w_i(\overline{\boldsymbol{x}}) = 1$. There is much space for designing weights as long as they reflect the effectiveness of a certain iteration in sampling a specific path. Generally, for each path $\overline{\boldsymbol{x}}$, the combination weight $w_i(\overline{\boldsymbol{x}})$ should be larger in the iteration with a higher sampling density $p_i(\overline{\boldsymbol{x}})$ or more samples. It could also be guided by auxiliary information, such as the overall variance of a technique [Grittmann et al. 2019]. Below, we first justify the general advantage of employing path-level reweighting.

Benefits of path-level reweighting. Compared to image-level combination methods, the primary benefit of our proposed reweighting lies in its ability to stabilize the values (i.e., importance weights) of individual samples. Since guiding distributions are reconstructed

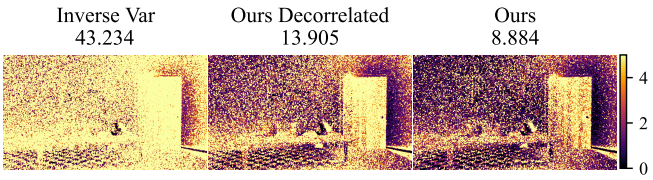

Fig. 5. Validation of the benefit of our path-level reweighting without correlation through a decorrelated (unbiased) variant. We show the relative variance map and RelMSE at equal sample rates (32 spp).

from a finite number of samples, some regions are often poorly fitted, especially in early iterations. An example is shown in Fig. 3. As seen, fitted distributions may even miss specific target distribution patterns, leading to an extremely small probability density $p_i(\overline{\boldsymbol{x}})$ while $f(\overline{\boldsymbol{x}})$ is still large. Consequently, samples in these regions will receive excessively high importance weights $f(\overline{\boldsymbol{x}})/p_i(\overline{\boldsymbol{x}})$, which would introduce high variance when combined into the final estimation.

Fortunately, a proper path-level reweighting scheme could capture specific iterations where some regions are poorly fitted, assigning them lower weights locally to mitigate the high variance they inject into the final estimation. This could be done by assigning lower $w_i(\overline{\boldsymbol{x}})$ to iterations with relatively lower $p_i(\overline{\boldsymbol{x}})$. As showcased in Fig. 4, many samples with high importance weights are weighted down (below the dashed line) after our reweighting process. Conversely, samples with low importance weights are mostly weighted up (above the dashed line). As a result, this reweighting process stabilizes the importance weights of samples without modifying the sampling process, effectively reducing the noise level.

Nevertheless, a unique issue in path guiding lies in the interdependence between distributions and samples. As weights are determined from multiple distributions fitted from samples, the weight of a particular sample indirectly depends on other samples. While this correlation could potentially decrease noise levels[1], it introduces bias into the estimation.

Validation and discussion. To discern how much variance reduction is coming from this correlation, we develop an impractical variant with reduced correlation. We splat half of the samples into the image and the other half into the distribution. The number of passes[2] for each iteration remains the same as the regular variant, but the samples per pass are doubled to ensure that the total number of samples in the image and distribution matches that of the regular variant. Fig. 5 demonstrates that this variant yields results with higher variance than the correlated variant but much lower than the baseline. This confirms that only a small portion of variance reduction stems from correlations.

[1]For instance, let $\overline{\boldsymbol{X}}_i$ denote the set of samples generated in the i-th iteration. If $\overline{\boldsymbol{X}}_{i-1}$ includes a sample with high importance weight (i.e., it is rarely found but contributes significantly), p_i will feature a peak in this region. Consequently, when reweighting samples using a mixture comprising p_{i-1} and p_i, the aforementioned high-weight samples will receive an excessively low weight.

[2]Each guiding iteration contains one or multiple passes, while in each pass, the number of samples per pixel is a constant (2 in our experiments).

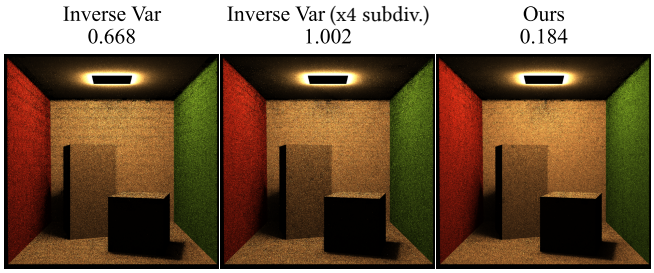

Fig. 6. Our distribution mixture makes it possible to use a 4× finer spatial subdivision to eliminate the striped artifacts due to spatial subdivision, while the conventional approach yields higher variance. Stochastic spatial filtering is enabled. We render direct illumination only for clearer differences. All methods take 72 spp in total using the same sample allocation.

Table 1. Our method is robust to sample allocation across iterations. We show the RelMSE of renderings. Our method demonstrates a more pronounced advantage when samples receive a finer division. Addition and multiplication represent concatenation and repeatition, respectively.

Allocation	Cbox (32 spp)			Ajar (32 spp)		
	I. Var	Ours	(Ratio)	I. Var	Ours	(Ratio)
$[1, 2, 4, 9]$	2.55	0.77	3.31×	14.61	3.28	4.46×
$[1] \times 4 + [1, 2, 4, 5]$	2.86	0.53	5.40×	11.69	2.61	4.48×
$[1] \times 8 + [1, 2, 5]$	3.52	0.43	8.17×	16.30	2.00	8.17×
$[1] \times 16$	4.04	0.42	9.71×	18.51	1.79	10.36×

3.2 Variance reduction potentials via sample division

Existing approaches strive to reduce the noise level of distributions and improve the robustness by keeping larger spatial regions (including 4,000-32,000 samples typically) when performing subdivisions and increasing the number of samples per iteration. However, these strategies come with limitations. Sharing distributions over a large spatial region diminishes accuracy, especially when dealing with scenes characterized by strong spatio-directional correlation [Dodik et al. 2022]. Moreover, increasing the number of samples per iteration could slow down fitting [Ruppert et al. 2020].

By sharing information across iterations, our method intrinsically enhances robustness, enabling a finer subdivision of samples both spatially and across iterations. In particular, when reducing the threshold of spatial subdivisions, our approach produces fewer artifacts attributable to parallax and subdivisions, which results in reduced overall variance. At the same time, conventional guiding could suffer from increased variance. An example is shown in Fig. 6, where the traditional approach already employs spatial filtering.

Similarly, allocating samples over iterations is also an important strategy for path guiding. The number of samples is often doubled with each iteration to ensure the quality of distributions, but this leads to a long turnaround time, which slows the convergence in difficult scenes [Ruppert et al. 2020]. By reweighting samples considering their probability density in each iteration, our combination could still produce low-variance results even if some paths are insufficiently sampled in certain iterations. This makes guiding

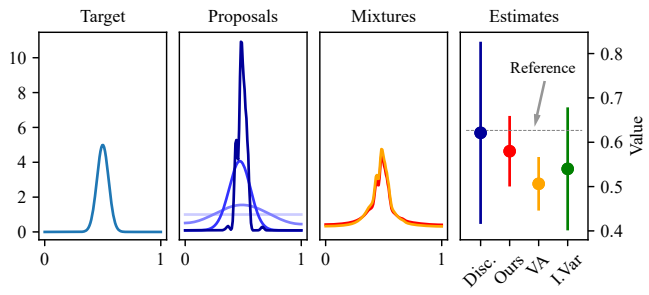

Fig. 7. A 1D example of the mixture-based reweighting scheme. We show iteratively learned proposals (sequentially from light to dark), and mixture distributions without variance awareness (red, Ours) and with variance awareness (yellow, VA). In the right-most subplot, we show the estimation mean and standard derivation to illustrate the variance and bias.

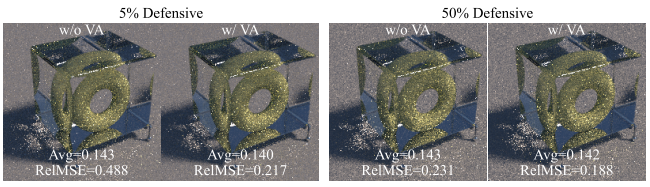

Fig. 8. We render a torus positioned on the ground under sunlight. Since sunlight can be quite accurately fitted by guiding distributions, variance awareness proves advantageous in diffuse regions (e.g., the upper left of the image) when employing minimal (5%, left) defensive sampling. However, this advantage is less visible in other regions or when employing a standard amount (50%, right) of defensive sampling. Through the average intensity (Avg), we observe that variance awareness leads to slightly more bias.

algorithms more robust to sample division, enabling a wide range of allocation schemes, as demonstrated in Table 1.

3.3 Mixture-based weighting schemes

Until now, our discussion is under a general setting that the combination weight $w_i(\overline{x})$ should be larger in the iteration that has a higher sampling density $p_i(\overline{x})$ and more samples. This is compatible with a variety of designs of weighting schemes.

A simple and effective choice is the balance heuristic [Veach and Guibas 1995b], which leads to the following weighting factor:

$$w_i(\overline{x}) = \frac{n_i p_i(\overline{x})}{\sum_{k=1}^{M} n_k p_k(\overline{x})}. \tag{6}$$

This weighting scheme is used by adaptive multiple importance sampling (AMIS) [Cornuet et al. 2009], proved to have the potential to leverage the most efficient proposal distribution as the poorly performing distributions are gradually phased out as M increases. This process can also be understood as if the samples were drawn from a mixture of distributions over all iterations. Moreover, within the framework of path guiding, spatial hierarchies usually undergo iterative refinement [Dodik et al. 2022; Müller et al. 2017; Ruppert et al. 2020]; thus, initial iterations will generate more coarse distributions while later iterations capture finer details but may risk over-fitting. In Fig. 7, we present a 1D example where the distribution of each iteration complements the others, enabling the utilization of a more robust distribution to determine importance weights. Moreover, an important advantage of this reweighting scheme is its provable consistency. We discuss it in the supplemental document.

Employing variance estimates. In certain cases, the above weighting scheme may yield higher variance compared to the inverse-variance-weighted combination, particularly when the last iterations accurately fit the target integrand, which, ideally, already produces zero-variance estimates. In this case, incorporating early distributions into the mixture could increase variance.

Inspired by how variance estimates improve the balanced-heuristic MIS [Veach and Guibas 1995b], we can similarly introduce a per-pixel variance-aware factor [Grittmann et al. 2019], the ratio of the second moment $u_{2,i}$ to the sample variance σ_i^2 for each iteration i. Then, each distribution is weighted accordingly:

$$w_i(\overline{x}) = \frac{n_i v_i p_i(\overline{x})}{\sum_{k=1}^{M} n_k v_k p_k(\overline{x})}, \qquad v_i = \frac{u_{2,i}}{\sigma_i^2}. \tag{7}$$

Nevertheless, it is not theoretically guaranteed to achieve consistent convergence when variance factors are used, and we also empirically observe more bias. Furthermore, as showcased in Fig. 8, the proposal distribution hardly fits the full integrand accurately in practice, which leads to a small variance factor (i.e., less than 3 in our tests), which limits the potential variance reduction. Therefore, we do not employ variance awareness in our experiments.

3.4 Acceleration and implementation

The computational and storage overhead is crucial to enable practical usage of path-level reweighting, which necessitates special care when implementing the sample reweighting process. In particular, to reweight the samples generated during training, we store them in memory. Once the proposal distribution of the final iteration is determined, these stored samples are reiterated. For each sample, we calculate the weights as per Eq. (6), where each probability density is expanded as in Eq. (5). Notably, the geometric terms cancel out, simplifying the computation to the product of a series of directional probability densities. However, a straightforward implementation necessitates $(N-1)M$ queries for probability densities. Considering the average computational cost for querying spatial and directional structures as C_S and C_D respectively, the overall time complexity amounts to $O((C_S + C_D)NM)$.

In practice, especially for large scenes, C_S can significantly outweigh C_D. To this end, we retain pointers to historical directional distributions within leaf nodes of the spatial hierarchy. After subdivisions of spatial nodes, only pointers are copied to avoid redundancy; new nodes may thus point to the same historical distribution. This optimization reduces the time complexity to $O(C_S N + C_D NM)$, almost halving the computational overhead.

For each path sample, we store its contribution and the positions of these N vertices. The PDF of defensive sampling and the direction at the last vertex are also required. We use single-precision float numbers (except for the direction, which is compressed into two 16-bit fixed-point numbers) and three channels for the spectrum. Hence, the N vertices require $4N$ float numbers while

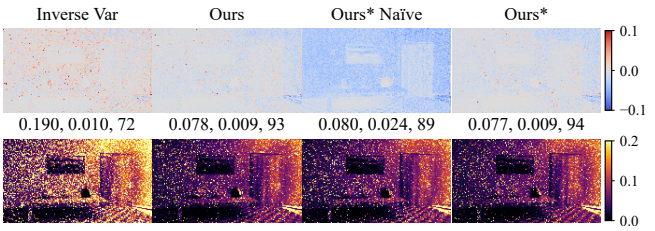

Fig. 9. The relative bias and relative squared error map of rendering the Ajar scene using 256 spp. We show RelMSE, relative mean absolute bias, and rendering time (seconds), respectively. Using a constant 500 MB sample storage, reweighting using partial mixtures (Ours*) performs much better than not reweighting low-value samples (Naïve) and enjoys comparable variance reduction and bias with storing all samples (Ours) using 973 MB.

the spectrum requires 3 float numbers, so a path costs $16(N+1)$ bytes of storage. Here, we only need to store samples with non-zero contributions; samples with zero contributions are directly splatted to the image. Nevertheless, the extra storage required by our method grows steadily as the number of samples increase.

3.5 Specifying a constant sample storage

The storage of path samples remains an inherent challenge of path-level reweighting. In practice, this can be alleviated by focusing more on the samples with higher importance weights[3]. In particular, we maintain a buffer with a constant size specified by users as a hyperparameter, which we set to 500 MB by default. When new samples are produced, we insert them into the buffer, a binary heap with importance weights as keys. Once the memory limit is exceeded, we remove samples with the lowest importance weights and simultaneously splat them to the image.

Unfortunately, when those samples are removed from the buffer, the distribution of the last few iterations is not generated, thus, the full mixture is unavailable. A simple choice is not to reweight any sample that is splatted during the rendering process. Since the aforementioned fact that many samples with high importance weights are weighted down and vice versa, this could lead to additional bias.

Our solution is still reweighting those samples using a partial mixture of distributions generated till the current (i-th) iteration:

$$w_i(\overline{x}) = \frac{n_i p_i(\overline{x})}{\sum_{k=1}^{i} n_k p_k(\overline{x})} \frac{\sum_{k=1}^{i} n_k}{\sum_{k=1}^{M} n_k}. \tag{8}$$

The performance of this method depends on how similar the partial mixture and the full mixture are. Despite without theoretical guaranteed consistency, we found that reweighting with a partial mixture performs well in variance reduction, bias, storage, and computational overhead. As shown in Fig. 9, the bias using a partial mixture (Ours*) is smaller than not reweighting any early-splatted samples (Ours* Naïve) and identical to using unlimited storage (Ours). This suggests that specifying a constant sample storage is a good choice for practical scenarios when memory is a bottleneck.

[3]Inspired by density-based outlier rejection, we also evaluated choosing the path samples with the largest distances to their neighbors. We found the difference between our method and choosing the highest importance weight samples is not visible.

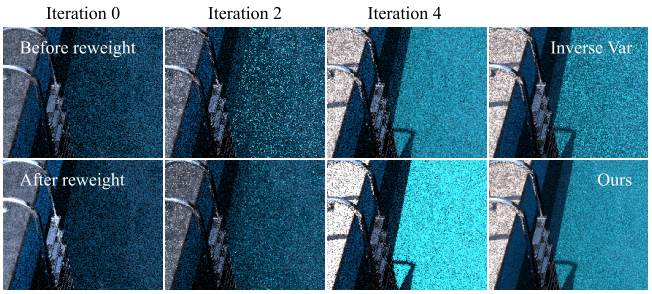

Fig. 10. In a pool scene with still water, we show the contribution of each iteration before and after reweighting. The sum of these partial renderings weighted by the number of samples yields the final result. The illumination from the sun is only well-fitted in the last iteration. Our reweighting effectively makes use of the well-sampled region from each iteration. For example, the ladder contributes more in early iterations, while the underwater region is weighted more in later iterations.

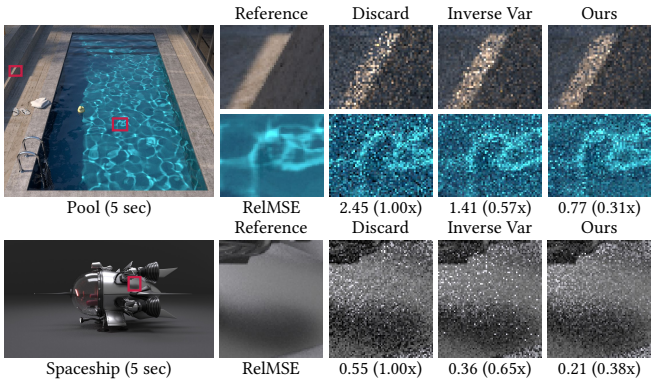

Fig. 11. Equal-time comparison with discard training samples [Müller et al. 2017] and image-level inverse-variance weighting [Müller 2019].

4 RESULTS

We have implemented our combination method upon Practical Path Guiding (PPG) [Müller et al. 2017] in Mitsuba 0.6 [Jakob 2010] and enable the stochastic spatio-directional splatting [Müller 2019]. For quantitative comparisons, we use Relative Mean Squared Error (RelMSE) as error metrics, with ϵ being 0.01. We also show Symmetric Mean Absolute Percentage Error (SMAPE) in some experiments. For our method, we set the spatial subdivision threshold to 1000 in the visual comparison of the Pool and Glossy Kitchen scene to validate the robustness of our method when using a finer subdivision while using 4000, the same as the original PPG, for other figures. Since computing pixel variance estimates require at least two samples per pixel, we use 2 spp per pass. Unless otherwise noted, the samples per iteration are doubled, but the first 4 iterations are allocated 2 spp to accelerate the adaptation of spatial structure following [Müller 2019]. For a fair comparison, both the inverse variance and our reweighting combine all images. Training is disabled in the final iteration of guiding. We disable Russian roulette and next-event estimations. Timings are done with a 24-core i9-13900KF processor.

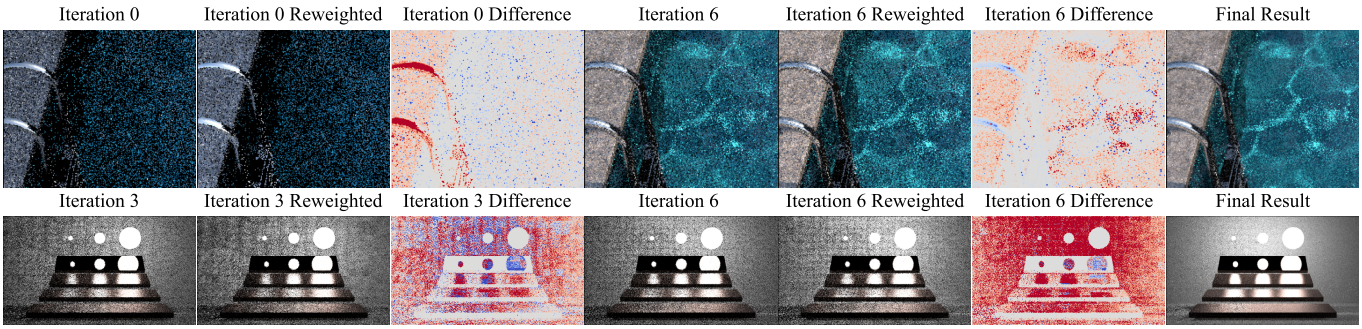

Fig. 12. Visualization of the contribution of each iteration *before and after reweighting*. The difference image illustrates that our reweighting procedure decreases the intensity of pixels colored in blue, while increasing the intensity of pixels colored in red.

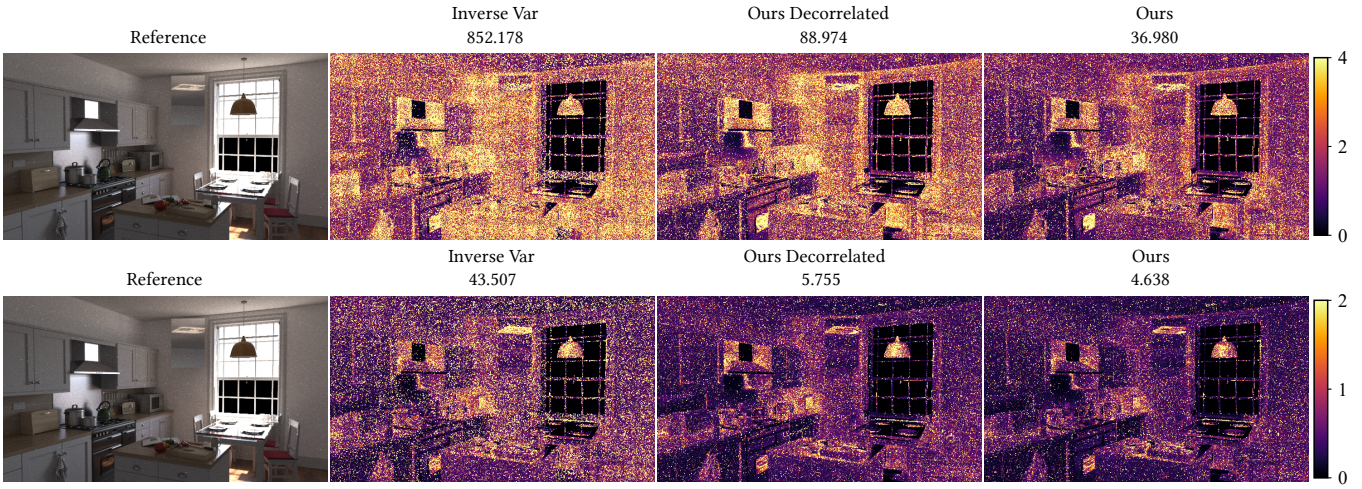

Fig. 13. Validation of the benefit of path-level reweighting without correlation through a decorrelated variant on the Kitchen scene at 24 spp and 128 spp, which includes challenging light paths. Only in this experiment do we enable Russian roulette, which start from the 5th bounce.

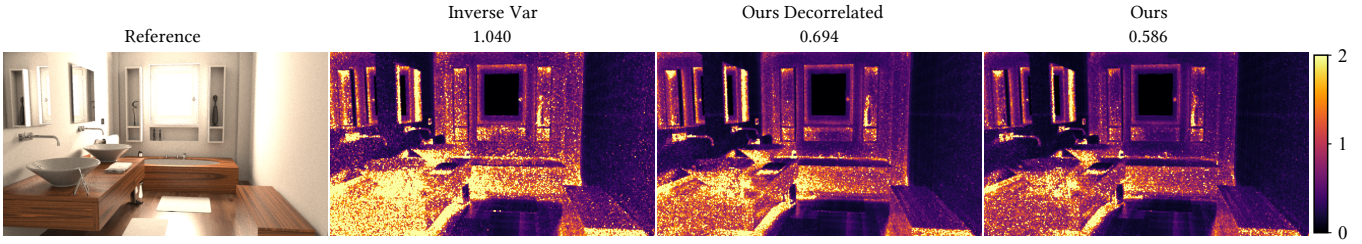

Fig. 14. Validation of the benefit of path-level reweighting on the bathroom scene at 24 spp, where the radiance field is relatively low-frequency.

Validation of the contribution of each iteration. In Figs. 10 and 12, we visualize the relative contribution of each iteration divided by the number of samples. Certain regions of the scene, such as those well handled by BSDF sampling, receive more contributions from early iterations. Conversely, features that require numerous iterations to fit, such as the caustics in the pool, are predominantly contributed by the last few iterations. Moreover, in regions where excessive variance arises from distribution over-fitting, as exemplified in the second scene of Fig. 12, our method adeptly attenuates their impact, thereby combining into a low-variance final result.

Comparison with inverse-variance weighting. Due to the instability and bias that could be introduced by per-pixel variance-based weighting (even when averaged per block or filtered as illustrated in Fig. 16), conventional practice dictates their averaging over the whole image [Müller 2019]. However, this fails to capture

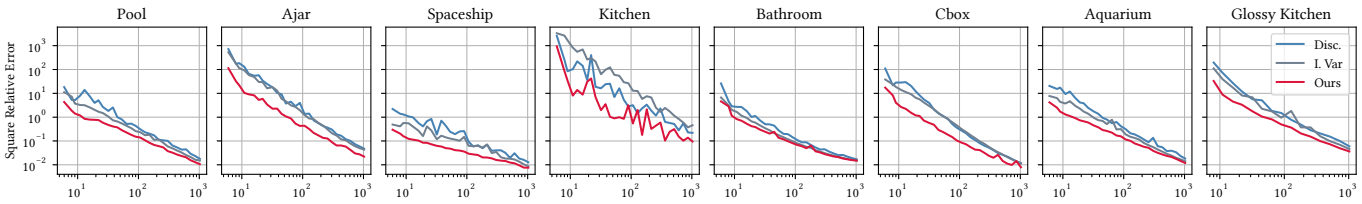

Fig. 15. Convergence curves of various combinations techniques, including discarding training samples, inverse-variance-weighted combination, and our path-level reweighting. We use a 500 MB constant sample storage. We show RelMSE (versus spp) averaged over three independent runs.

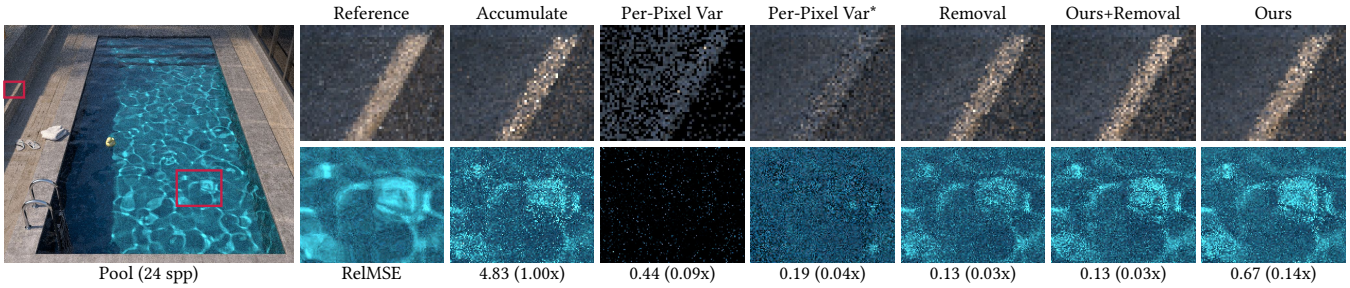

Fig. 16. Comparison with other pixel-level combination techniques: simply accumulating all samples, inverse-variance-weighting using per-pixel estimates (Per-Pixel Var) and filtered ones (Per-Pixel Var*), and accumulating using outlier removal [Zirr et al. 2018].

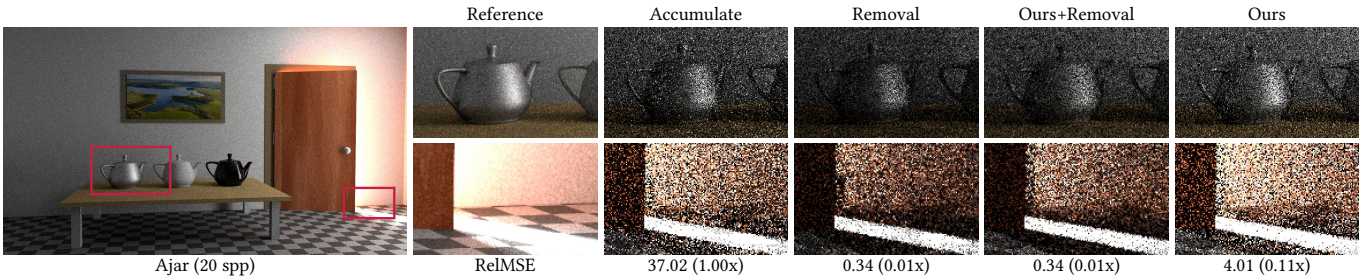

Fig. 17. We compare our method with simply accumulating all samples and accumulating using outlier removal [Zirr et al. 2018], which leads to much more bias than our reweighting. Our method could also be used together with outlier removal to reduce bias.

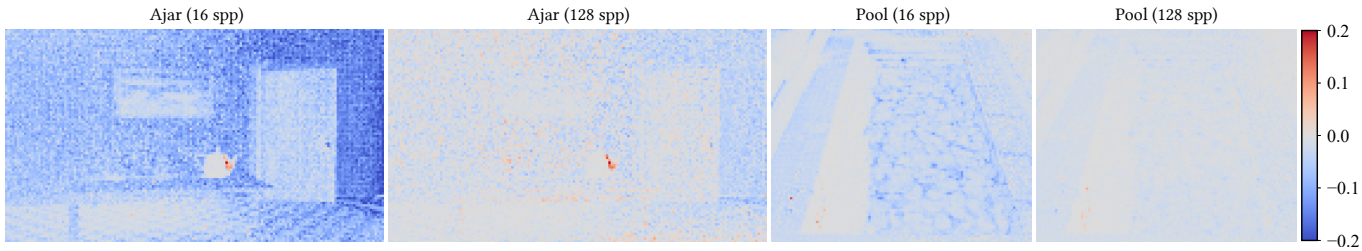

Fig. 18. Visualization of the relative bias computed by averaging over 256 independent runs and down-sampled to reduce noise. We observe that the bias progressively diminishes as the number of samples increases, which empirically validates the consistency of our method.

the varying convergence rates across different regions within path space. As a result, the combined images suffer from high variance in some regions, as shown in Fig. 11. When employing our path-level reweighting, paths that are insufficiently sampled in certain iterations are weighted down (Fig. 4), which reduces the variance they introduce. Consequently, as demonstrated in the visual comparison in equal time (Fig. 11), our approach often leads to more noise reduction than inverse-variance weighting.

Impact of scene complexity and budgets. Additionally, our method exhibits more pronounced benefits in scenes characterized by

Table 2. Rendering statistics of our experiments (24 spp). We report the ratio of variance reduction and rendering time compared to the inverse-variance-weighted combination. We show the storage overhead of samples (S) and distributions (D), respectively. We also report the original memory footprint (O) of guiding distributions without history. The last column (NZ) represents the percentage of samples (over the entire guiding process) that have a non-zero contribution. The resolution is 1000×1000 for Cbox, 200×200 for Torus, 1440p for Glossy Kitchen, and 720p for other scenes. Our sample storage would grow as the resolution and spp increases.

Scene	Depth Max	RelMSE	Time/sec	Memory/MB S	D	O	NZ %
Aquarium	12	0.345 (3.0×)	18 (1.13×)	359	160	49	14
Ajar	4	2.235 (6.8×)	11 (1.08×)	105	84	25	10
Bathroom	4	0.268 (1.2×)	17 (1.11×)	240	67	20	30
Cbox	3	0.477 (5.5×)	5 (1.03×)	58	65	20	6
Glossy Kitchen	10	1.564 (2.8×)	58 (1.16×)	716	465	143	11
Kitchen	10	6.438 (18.4×)	21 (1.16×)	353	169	51	26
Pool	4	0.342 (3.2×)	9 (1.17×)	321	35	11	57
Spaceship	10	0.041 (2.1×)	13 (1.41×)	625	70	22	90
Torus	8	0.203 (1.5×)	1 (0.97×)	32	2	1	82

challenging light paths. For example, in Fig. 14, the variance reduction is relatively marginal compared to scenes that include many hard-to-sample paths (e.g., Fig. 13). The benefit is also more significant when using a low sample budget in Fig. 13. These scenarios typically entail distributions with a higher noise level, where a mixture-based reweighting effectively reduces the noise in distributions, thus leading to more stable importance weights and lower variance.

Convergence plot. In Fig. 15, we investigate the convergence behavior when using different combination techniques, where our proposed reweighting consistently outperforms previous approaches. Moreover, the advantage is more evident with a smaller sample budget. With an increase in the number of samples, the distributions tend to be more accurate, which reduces the potential impact of sample combination.

Comparison with accumulation and outlier removal. In Fig. 17 and Fig. 16, we compare our method with directly accumulating all samples and an outlier removal [Zirr et al. 2018]. As seen, splatting all samples without reweighting often leads to excessively high variance since the initial iterations are extremely noisy. Utilizing outlier removal upon the accumulated samples [Reibold et al. 2018] helps mitigate problematic samples at the cost of visible bias. After all, outlier removal solely operates on the values (i.e., importance weights) of samples, whereas our reweighting leverages the information of distributions. Furthermore, our weights are normalized over iterations. This results in substantially lower bias compared to outlier removal. Additionally, our method can be used together with outlier removal, almost halving its bias in our experiments.

Bias evaluation. In Fig. 18, we evaluate the bias of our approach by averaging the renderings from many independent runs and contrasting them with the reference. Generally, the bias is in the form of energy loss attributed to the aforementioned correlations across iterations. The bias remains significantly smaller compared to the noise level and diminishes gradually as the sample rates increase.

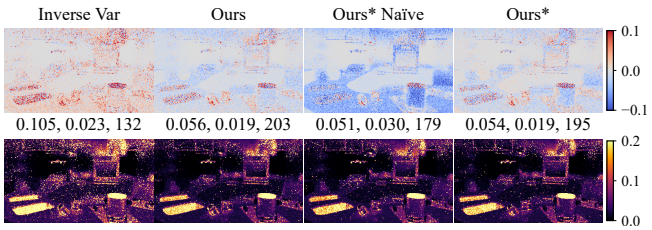

Fig. 19. The relative bias and error map of rendering the Glossy Kitchen scene using 512 spp. We show the RelMSE, rBias, and rendering time (seconds), respectively. Using a constant memory (Ours*) is comparable to unlimited storage (over 10 GB) in variance and bias. The naïve solution, which does not reweight early-splatted samples, introduces significant bias.

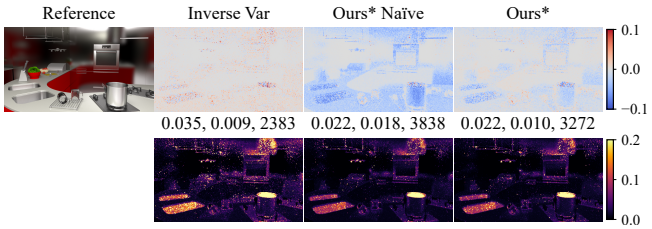

Fig. 20. Evaluating high-end rendering on the Glossy Kitchen scene using 1024 spp at 2560 × 1440 resolution. The numbers represent the RelMSE, rBias, and rendering time (seconds), respectively.

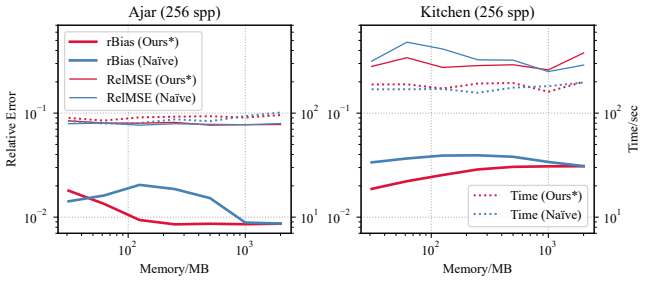

Fig. 21. The impact of the sample storage on the relative absolute bias, relative mean squared error, and rendering time. Reweighting using a partial mixture enjoys almost consistently lower bias than the naïve variant at the cost of slightly more computational overhead. The difference in RelMSE is not visible. As the storage goes to infinity, the method becomes consistent.

Performance analysis. We present the statistics of our experiments in Table 2. The storage overhead is generally around hundreds of megabytes in our test, which is acceptable in practical scenarios. The extra computational time majorly comes from the evaluation of weights, which is around 30% in our scenes and even neglectable in scenes where most paths miss the light source so the number of paths required to be reweighted is small.

Sample storage. In Figs. 9 and 19, we compare using constant sample storage against using unlimited ones. Simply not reweighting any low-contribution samples results in higher bias levels than reweighting all samples. Through still reweighting using a

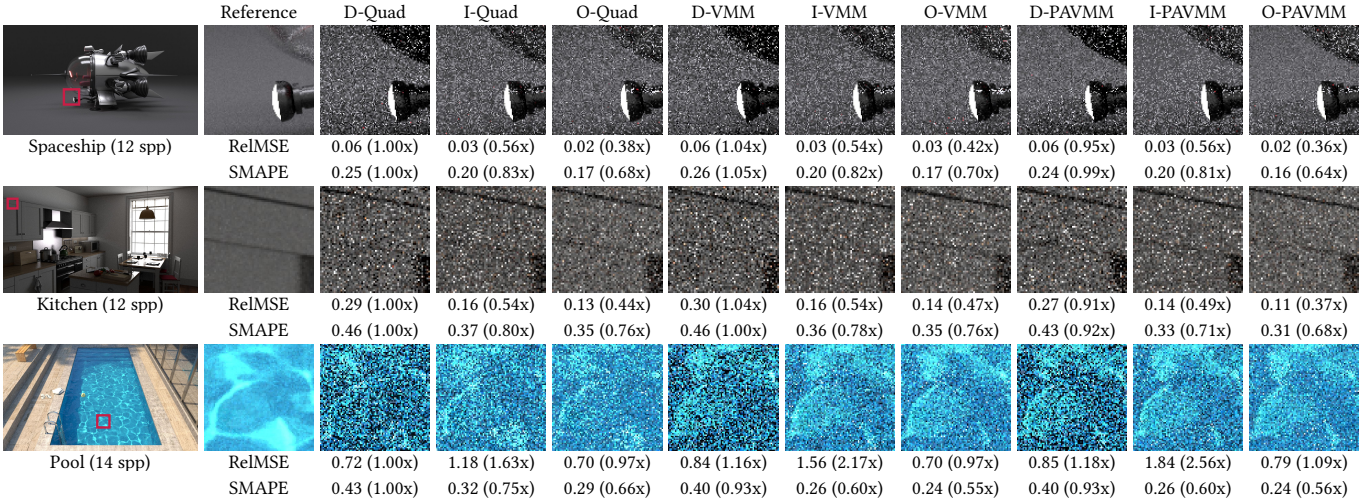

Fig. 22. Evaluation of our approach's generality on three distributions (quadtree, vMF mixture, and parallax-aware vMF mixture) from the OpenPGL library [Herholz and Dittebrandt 2022]. We compare discarding training samples (D), inverse-variance weighting (I), and our path-level reweighting (O).

partial mixture, we use constant sample storage to produce results identical to unlimited storage in our test scenes, consistently better than not reweighting any low-contribution samples as shown in Fig. 21. We also showcase a high-resolution example in Fig. 20.

Distribution models. Our proposed method generalizes over various distribution models, as shown in Fig. 22 on three distributions in Intel's OpenPGL library [Herholz and Dittebrandt 2022]. Empirically, our method produces the lowest variance consistently.

5 CONCLUSION

The rigid division between training and rendering presents numerous challenges in the development of guiding algorithms. Unlike conventional approaches that address this issue simply through a weighted summation of images, our method provides a thorough reassessment of the whole sample set. We assign weights to individual path samples, reflecting the quality disparities within the distributions. By evaluating how these samples are generated and identifying well-sampled regions against potential flaws, we endeavor to optimize sample utilization while mitigating risks of high variance, outliers, or biasedness. This suggests good potential for unbounded variance reduction and yields robust, practical performance with provably consistent convergence under certain conditions.

Discussions and future works. The main limitation of our method is the storage and computation cost, which can be prohibitive for some uses, especially when the resolution and number of samples are high. In such cases, the computational overhead is more severe, so the improvement is minimal in an equal-time setting. We show examples in the supplemental document. Still, certain combinations with simpler strategies may be helpful. For instance, one can apply our reweighting during the early iterations and switch to inverse-variance weighting after the distribution is converged later.

Besides, the storage of path sample may also create challenges for GPU implementations [Lu et al. 2024]. Also, for neural methods [Dong et al. 2023; Huang et al. 2024], the distribution storage could be more costly. Note that our optimization for spatial hierarchy mainly works for a regular kd-tree [Fan et al. 2023; Müller et al. 2017]. For other structures [Dodik et al. 2022; Reibold et al. 2018], a full copy of historical distributions may be required.

Additionally, next-event estimation (NEE) is currently disabled in our experiments, but enabling it is straightforward. In particular, assuming we use a balanced heuristic, the PDF would be changed to a mixture of light sampling, BSDF sampling, and (averaged) guided sampling. However, the implementation complexity would increase due to substantial changes required in the path storage. Future works could extend our method to more complicated guiding approaches on complex rendering algorithms including those with light sampling, bidirectional connections, and vertex merging.

Lastly, future opportunities for variance reduction still exist. While reweighting during training (as in the original AMIS [Cornuet et al. 2009]) could further reduce variance, it increases overhead and additional bias. Importantly, the consistency proof holds only when reweighting is not performed during training. Thus, we only reweight when generating the final estimation. Yet, its variance reduction potential may deserve future research. Also, our balanced-heuristic weighting may not be the best weighting scheme in terms of variance reduction or bias level, as demonstrated in Fig. 8.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable suggestions. We thank Pengpei Hong and Yiyang Sun for valuable discussions at the early stage of this project. This work was supported by the National Natural Science Foundation of China (No.

61972194 and No. 62032011) and the Natural Science Foundation of Jiangsu Province (No. BK20211147).

REFERENCES

Olivier Cappé, Randal Douc, Arnaud Guillin, Jean-Michel Marin, and Christian P. Robert. 2008. Adaptive Importance Sampling in General Mixture Classes. *Statistics and Computing* 18, 4 (Dec. 2008), 447–459.

Jean-Marie Cornuet, Jean-Michel Marin, Antonietta Mira, and Christian Robert. 2009. Adaptive Multiple Importance Sampling. *Scandinavian Journal of Statistics* 39 (07 2009).

Mikhail Derevyannykh. 2022. Real-Time Path-Guiding Based on Parametric Mixture Models. (2022).

Ana Dodik, Marios Papas, Cengiz Öztireli, and Thomas Müller. 2022. Path Guiding Using Spatio-Directional Mixture Models. *Computer Graphics Forum* 41, 1 (Feb. 2022), 172–189.

Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. *Conference Proceedings* (2023).

Yousef El-Laham, Luca Martino, Victor Elvira, and Monica F. Bugallo. 2019. Efficient Adaptive Multiple Importance Sampling. *2019 27th European Signal Processing Conference (EUSIPCO)* (Sept. 2019), 1–5.

Víctor Elvira and Luca Martino. 2021. Advances in Importance Sampling. (Feb. 2021).

Zhimin Fan, Pengpei Hong, Jie Guo, Changqing Zou, Yanwen Guo, and Ling-Qi Yan. 2023. Manifold Path Guiding for Importance Sampling Specular Chains. *ACM Trans. Graph.* 42, 6, Article 257 (dec 2023), 14 pages.

Zhimin Fan, Pengcheng Shi, Mufan Guo, Ruoyu Fu, Yanwen Guo, and Jie Guo. 2024. Conditional Mixture Path Guiding for Differentiable Rendering. *ACM Trans. Graph.* 43, 4, Article 48 (July 2024), 11 pages.

Luca Fascione, Johannes Hanika, Rob Pieké, Ryusuke Villemin, Christophe Hery, Manuel Gamito, Luke Emrose, and André Mazzone. 2018. Path Tracing in Production. In *ACM SIGGRAPH 2018 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, Article 15, 79 pages.

Pascal Grittmann, Iliyan Georgiev, and Philipp Slusallek. 2021. Correlation-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms. *Computer Graphics Forum* 40, 2 (2021), 231–238.

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Transactions on Graphics* 38, 6 (Nov. 2019), 152:1–152:9.

Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. Efficiency-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms. *ACM Trans. Graph.* 41, 4 (July 2022).

Vlastimil Havran and Mateu Sbert. 2014. Optimal Combination of Techniques in Multiple Importance Sampling. In *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI '14)*. Association for Computing Machinery, New York, NY, USA, 141–150.

Sebastian Herholz and Addis Dittebrandt. 2022. Intel® Open Path Guiding Library. http://www.openpgl.org.

Sebastian Herholz, Oskar Elek, Jiří Vorba, Hendrik Lensch, and Jaroslav Křivánek. 2016. Product Importance Sampling for Light Transport Path Guiding. *Computer Graphics Forum* 35, 4 (2016), 67–77.

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik P. A. Lensch, and Jaroslav Křivánek. 2019. Volume Path Guiding Based on Zero-Variance Random Walk Theory. *ACM Transactions on Graphics* 38, 3 (June 2019), 1–19.

Jiawei Huang, Akito Iizuka, Hajime Tanaka, Taku Komura, and Yoshifumi Kitamura. 2024. Online Neural Path Guiding with Normalized Anisotropic Spherical Gaussians. *ACM Trans. Graph.* 43, 3, Article 26 (April 2024), 18 pages.

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

James T. Kajiya. 1986. The Rendering Equation. *Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86* (1986), 143–150.

A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols. 2015. The Path Tracing Revolution in the Movie Industry. In *ACM SIGGRAPH 2015 Courses* (Los Angeles, California) *(SIGGRAPH '15)*. Association for Computing Machinery, New York, NY, USA, Article 24, 7 pages.

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Transactions on Graphics* 38, 4 (Aug. 2019), 1–14.

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

Jean-Michel Marin, Pierre Pudlo, and Mohammed Sedki. 2019. Consistency of Adaptive Importance Sampling and Recycling Schemes. *Bernoulli* 25, 3 (Aug. 2019).

Thomas Müller. 2019. "Practical Path Guiding" in Production. In *ACM SIGGRAPH Courses: Path Guiding in Production, Chapter 10* (Los Angeles, California). ACM, New York, NY, USA, 18:35–18:48.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum* 36, 4 (July 2017), 91–100.

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM Transactions on Graphics* 38, 5 (Oct. 2019), 1–19.

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).

Alexander Rath, Ömercan Yazici, and Philipp Slusallek. 2023. Focal Path Guiding for Light Transport Simulation. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 30, 10 pages.

Florian Reibold, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2018. Selective Guided Sampling with Complete Light Transport Paths. *ACM Transactions on Graphics* 37, 6 (Dec. 2018), 1–14.

Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. 2020. Robust Fitting of Parallax-Aware Mixtures for Path Guiding. *ACM Transactions on Graphics* 39, 4 (Aug. 2020).

Vincent Schüßler, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2022. Path Guiding with Vertex Triplet Distributions. *Computer Graphics Forum* 41, 4 (July 2022), 1–15.

Yusuke Tokuyoshi, Shinji Ogaki, and Schoellhammer Sebastian. 2010. Final Gathering Using Adaptive Multiple Importance Sampling. In *ACM SIGGRAPH ASIA 2010 Posters (SA '10)*. Association for Computing Machinery, New York, NY, USA, 1.

Eric Veach. 1997. Robust Monte Carlo Methods for Light Transport Simulation.

Eric Veach and Leonidas Guibas. 1995a. Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques*, Georgios Sakas, Stefan Müller, and Peter Shirley (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 145–167.

Eric Veach and Leonidas J. Guibas. 1995b. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 419–428.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses*. ACM, Los Angeles California, 1–77.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-Line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Transactions on Graphics* 33, 4 (July 2014), 1–11.

Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. 2020. Continuous Multiple Importance Sampling. *ACM Transactions on Graphics* 39, 4 (Aug. 2020), 136:136:1–136:136:12.

Tobias Zirr, Johannes Hanika, and Carsten Dachsbacher. 2018. Reweighting Firefly Samples for Improved Finite-Sample Monte Carlo Estimates. *Computer Graphics Forum* 37, 6 (Sept. 2018), 410–421.