Efficiency-aware multiple importance sampling for bidirectional rendering algorithms

PASCAL GRITTMANN, Saarland University, Germany
ÖMERCAN YAZICI, Saarland University, Germany
ILIYAN GEORGIEV, Autodesk, United Kingdom
PHILIPP SLUSALLEK, Saarland University, Germany and DFKI, Germany

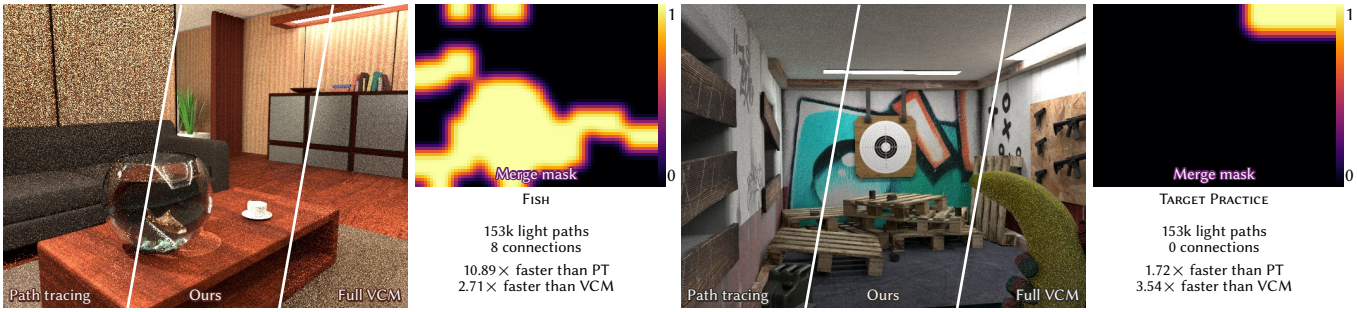

Fig. 1. Two scenes that are challenging to render efficiently without user guidance: The caustics and strong indirect illumination in the Fish scene (left) resolve much faster with a bidirectional method such as VCM. But the simpler Target Practice scene (right) renders 2× slower with VCM than with forward path tracing. Our method renders both scenes efficiently by automatically setting the number of light subpaths to trace, the number of bidirectional connections to make, and in which pixels to perform photon density estimation (the 'merge mask' specifies a probability for performing a photon lookup in a pixel).

Multiple importance sampling (MIS) is an indispensable tool in light-transport simulation. It enables robust Monte Carlo integration by combining samples from several techniques. However, it is well understood that such a combination is not always more efficient than using a single sampling technique. Thus a major criticism of complex combined estimators, such as bidirectional path tracing, is that they can be significantly less efficient on common scenes than simpler algorithms like forward path tracing. We propose a general method to improve MIS efficiency: By cheaply estimating the efficiencies of various technique and sample-count combinations, we can pick the best one. The key ingredient is a numerically robust and efficient scheme that uses the samples of one MIS combination to compute the efficiency of multiple other combinations. For example, we can run forward path tracing and use its samples to decide which subset of VCM to enable, and at what sampling rates. The sample count for each technique can be controlled per-pixel or globally. Applied to VCM, our approach enables robust rendering of complex scenes with caustics, without compromising efficiency on simpler scenes.

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

Authors' addresses: Pascal Grittmann, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany, grittmann@cg.uni-saarland.de; Ömercan Yazici, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany, yazici@cg.uni-saarland.de; Iliyan Georgiev, Autodesk, London, United Kingdom, iliyan.georgiev@autodesk.com; Philipp Slusallek, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany and DFKI, Saarland Informatics Campus, Saarbrücken, Germany, philipp.slusallek@dfki.de.

0730-0301/2022/7-ART80
https://doi.org/10.1145/3528223.3530126

Additional Key Words and Phrases: ray tracing, global illumination, Monte Carlo, multiple importance sampling

ACM Reference Format:
Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. Efficiency-aware multiple importance sampling for bidirectional rendering algorithms. *ACM Trans. Graph.* 41, 4, Article 80 (July 2022), 12 pages. https://doi.org/10.1145/3528223.3530126

1 INTRODUCTION

Monte Carlo integration has firmly established itself as the standard approach for computing global illumination, thanks to its ability to achieve high levels of accuracy and realism. Many Monte Carlo rendering algorithms have been developed over the past three decades, with varying degree of sophistication and capability to reproduce complex illumination effects. The practical utility of such an algorithm depends crucially on two factors: (1) its robustness in handling diverse scenes and (2) its efficiency on typical scenes. The best known way to achieve robustness is to combine diverse sampling techniques through multiple importance sampling (MIS) [Veach and Guibas 1995b]. For example, the vertex connection and merging (VCM) algorithm comprises a large number of techniques, each specialized to efficiently capture a different illumination effect [Georgiev et al. 2012; Hachisuka et al. 2012]. While robust, such extensive combinations can incur significant computational overhead on typical scenes that contain only a subset of all possible effects. Achieving efficiency involves per-scene parameter tuning, e.g., the number of light subpaths or shadow-ray connections, or completely disabling redundant techniques. This in turn requires good understanding of the scene's lighting and the utility of every available technique. To avoid burdening users, modern renderers

often sacrifice robustness by defaulting to simple forward path tracing which can efficiently handle many common configurations but not complex indirect illumination, occlusion, and caustics [Křivánek et al. 2018].

Automatic parameter tuning has the potential to greatly improve the practical utility of comprehensive MIS strategies. To that end, we propose a simple and general method for optimizing sample counts in MIS. By taking samples from only one technique, we can cheaply estimate the efficiency of all available techniques and predict the best sample allocation for each. For example, from a single path-tracing invocation with one sample per pixel, our method can reliably determine if and how the given scene would benefit from bidirectional sampling. Based on that information, we can either enable such techniques or, if these are deemed inefficient, continue rendering using basic techniques, without ever incurring the overhead of the more sophisticated ones. Figure 1 demonstrates the effectiveness of our approach on the VCM algorithm, which employs one of the most complex MIS combinations. Our method can enable techniques on-demand, to achieve high efficiency without any user guidance. The net result is consistent and potentially substantial speed-up over both unidirectional path tracing and VCM with fixed parameters.

We focus on complex bidirectional algorithms like VCM. Our key contribution is an efficient estimation scheme that supports the high-dimensional path densities encountered in such algorithms (Section 3.3). The underlying ideas of our method are general and can be extended to other MIS applications. The supplemental document shows results for a simple direct illumination application. Source code is available on GitHub [Grittmann et al. 2022].

2 PREVIOUS WORK

There have been a few attempts at improving the sample allocation for MIS combinations. These have focused on comparatively simple low-dimensional applications, usually local sampling decisions for direct illumination or path guiding applications. Instead, our method is specifically designed to support complex high-dimensional applications like bidirectional algorithms.

Heuristics. Pajot et al. [2010] were the first to attack the problem of optimal sample allocation for estimating reflected radiance. They designed per-technique heuristics that can measure how "relevant" a technique is for a given configuration. For example, their heuristics for path guiding [Jensen 1995] assess whether the learned incident radiance distribution or the BSDF is more important at each point in the scene. However, this formulation cannot easily be generalized to other applications.

Variance estimates. As an alternative to a heuristic approach, the sample counts can be set based on variance estimates of individual techniques [Havran and Sbert 2014; Sbert et al. 2018b,a, 2016; Sbert and Havran 2017], but it has been shown that direct optimization of the combined MIS variance yields better results [Sbert et al. 2019].

Convex optimization. Three different approaches have been proposed to directly minimize the variance of a one-sample MIS estimator on-the-fly during rendering: Taylor approximation [Lu et al. 2013], Newton-Raphson root-finding [Sbert et al. 2019; Murray et al. 2020], and gradient descent [Müller 2019; Rath et al. 2020]. The optimized technique-selection probabilities can be also utilized in a multi-sample MIS combination whose variance is upper-bounded by the corresponding one-sample combination [Sbert et al. 2019]. In Appendix A we show how these approaches can be extended to take (linear) sampling cost into account and how the required derivatives can be computed for high-dimensional integrals in a numerically stable and efficient manner. However, convex optimization only works if the classic balance heuristic is used as the weighting function, which has been shown to be problematic for bidirectional algorithms [Grittmann et al. 2019, 2021].

Sample densities and weights. Not only the sample counts can be optimized in an MIS combination. Prior work has shown how to enhance the technique-weighting functions to better handle failure cases [Grittmann et al. 2019, 2021] and has even derived the optimal weights [Kondapaneni et al. 2019]. MIS has a close relationship to control variates [Owen and Zhou 2000; Kondapaneni et al. 2019], and it is possible to jointly optimize the control-variate coefficients and the sample counts [He and Owen 2014]. Sbert and Elvira [2022] provide a theoretical discussion of optimized weights and sample counts under various constraints and formulations. Optimizing the sampling densities themselves has also been investigated [Karlík et al. 2019]. Hachisuka et al. [2014] proposed to mutate the choice of sampling technique when using MIS for Markov chain Monte Carlo integration.

Bidirectional rendering. Optimal sample allocation for bidirectional rendering algorithms has not been thoroughly explored in previous work. One example is the number of light subpaths in photon mapping. Grittmann et al. [2018] have shown that optimizing this number can have a drastic impact on performance. They used a very simple heuristic to set it to the number of pixels that require photons. This approach works only because their emission-guiding approach focuses photons into important regions that are small in image space; it cannot be generalized easily. We replace this ad-hoc heuristic by a generic formulation. Another application is resampled connections in bidirectional path tracing [Popov et al. 2015; Nabata et al. 2020], where each camera-subpath vertex is connected to c light-subpath vertices. The choice of c impacts rendering performance, which is why Popov et al. [2015] manually selected a number for each scene. We show that this selection can be automated. In fact, we can jointly optimize the number of connections with the number of light paths along with the decision whether to perform photon mapping in each pixel.

3 OUR APPROACH

Our goal is to optimize the allocation of samples in MIS estimators, which has the potential to substantially improve their efficiency. We attack this problem via a simple brute-force scheme: We compute the efficiencies of a set of candidate sample-allocation strategies from a pilot estimation run that typically uses only one technique. (Our VCM application in Section 4 uses unidirectional path tracing as the pilot strategy.) Rendering then proceeds with the best strategy; optionally, the process can be refined iteratively.

In the remainder of this section, we formulate the problem (Section 3.1), describe our optimization algorithm (Section 3.2), and present an efficient scheme to compute the required quantities (Section 3.3). In Appendix A we show how our insights can be used to improve existing convex optimization approaches, for the special cases where the optimization objective remains convex.

3.1 Background and problem statement

We consider the definite integral I of a function $f\colon \mathcal{X} \to \mathbb{R}$ over some domain $\mathcal{X}$ and a multi-sample MIS estimator $\langle I\rangle_{\mathbf{n}}$ of the integral that combines samples from T techniques:

$$I = \int_{\mathcal{X}} f(x)\,\mathrm{d}x, \qquad \langle I\rangle_{\mathbf{n}} = \sum_{t=1}^{T}\sum_{i=1}^{n_t} w_{\mathbf{n},t}(x_{t,i})\frac{f(x_{t,i})}{n_t p_t(x_{t,i})}. \quad (1)$$

Here, n_t samples $x_{t,i}$ are drawn from technique t, and a weight $w_{\mathbf{n},t}(x_{t,i})$ is assigned to each. A provably good choice for the weighting functions $w_{\mathbf{n},t}$ is the balance heuristic [Veach and Guibas 1995b]. We consider its extension [Grittmann et al. 2019, 2021],

$$w_{\mathbf{n},t}(x) = \frac{c_t(x)n_t p_t(x)}{\sum_{t'} c_{t'}(x)n_{t'}p_{t'}(x)}, \quad (2)$$

that applies correction factors $c_t(x)$ to the effective sampling densities $n_t p_t(x)$, in order to better account for technique variances and correlations in their samples.

Strategy. We refer to the vector

$$\mathbf{n} = (n_1, \ldots, n_T) \quad (3)$$

as the *sample-allocation strategy*. Different strategies $\mathbf{n}$ yield different MIS estimators $\langle I\rangle_{\mathbf{n}}$. The best choice depends on the integrand and the sampling densities. In many cases it is beneficial to completely disable individual techniques by setting $n_t = 0$. For instance, photon mapping is of little use in scenes without caustics.

Efficiency. In rendering, the value of each pixel is a separate integral. Since our goal is to optimize the rendering efficiency of the whole image, we aim to minimize the product of total expected cost (i.e., expected render time) and total relative variance (i.e., expected error) across all integrals:

$$\arg\min_{\mathbf{n}}\, C(\mathbf{n}) \sum_{j=1}^{P} \frac{V[\langle I_j\rangle_{\mathbf{n}}]}{I_j^2}. \quad (4)$$

Here, P is the number of pixels and $V[\langle I_j\rangle_{\mathbf{n}}]$ is the variance of the jth pixel when using strategy $\mathbf{n}$. Normalizing each variance by the corresponding squared ground truth I_j^2 is optional; the resulting relative variance accounts for the varying magnitudes of the individual integrals (i.e., bright and dark pixels). $C(\mathbf{n})$ is the cost of all consumed samples across all pixels. It can be approximated with a heuristic, estimated from rendering statistics, or a mix of both. For example, our VCM application combines statistics of the average path lengths and a heuristic based on the number of rays.

Variance vs. second moment. The variance $V[\langle I_j\rangle_{\mathbf{n}}]$ of a multi-sample MIS estimator is the difference between the second moment $M[\langle I_j\rangle_{\mathbf{n}}]$ and a residual $r_{\mathbf{n}}$:

$$V[\langle I\rangle_{\mathbf{n}}] = M[\langle I\rangle_{\mathbf{n}}] - r_{\mathbf{n}}. \quad (5)$$

When using the extended balance heuristic (2), the second moment is the integral

$$M[\langle I\rangle_{\mathbf{n}}] = \int_{\mathcal{X}} \frac{f^2(x)\sum_t w_{\mathbf{n},t}(x)c_t(x)}{\sum_t c_t(x)n_t p_t(x)}\,\mathrm{d}x \quad (6)$$

which can be estimated efficiently, as we discuss in Section 3.3. The residual $r_{\mathbf{n}}$, however, is a sum of squared integrals, or even nested squared integrals if samples are correlated. In the context of VCM, the number of squared integrals is quadratic in the maximum path length, making the residual extremely expensive to estimate. Fortunately, the second moment is generally a good proxy for the variance: $V[\langle I\rangle_{\mathbf{n}}] \approx M[\langle I\rangle_{\mathbf{n}}]$. Veach and Guibas [1995b] used this approximation to show the variance guarantees of the balance heuristic. However, there are two cases where the approximation fails: when one technique has very low variance [Grittmann et al. 2019; Veach 1997], or when the samples are severely correlated [Grittmann et al. 2021]. We have found that using the second moment works well for our optimization. Finding an efficient approximation or estimation scheme for the residual term is likely to yield further improvement. Our simple search-based optimization, described in Section 3.2, readily supports such improvements. The supplemental document discusses the approximation error further.

Optimization objective. Our goal is to solve the slightly suboptimal but easier to handle optimization problem

$$\mathbf{n} = \arg\min_{\mathbf{n}} C(\mathbf{n}) \sum_{j=1}^{P} \frac{M[\langle I_j\rangle_{\mathbf{n}}]}{I_j^2}, \quad (7)$$

i.e., to find the sample-allocation strategy $\mathbf{n}$ that approximately maximizes the image-rendering efficiency by minimizing the work-normalized relative second moments.

3.2 Brute-force optimization

We employ a simple yet effective brute-force search to find the optimal strategy $\mathbf{n}$ that minimizes Eq. (7). The benefit of such a brute-force approach is that it does not require the objective to be convex, i.e., it supports arbitrary MIS weighting functions, cost functions, and additional constraints.

Candidates. We start by specifying a set of N promising candidate sample-allocation strategies $\{\mathbf{n}^1, \ldots, \mathbf{n}^N\}$. This can be done either by (regularly) sampling a range of allowed sample counts for each technique, or by applying prior knowledge to identify combinations that are likely to be efficient. Depending on the application, sample counts can be controlled per pixel and/or on an image level. For example, in VCM the number of light subpaths is specified only globally, while the number of connections can be set per pixel. A strategy $\mathbf{n} = (\mathbf{p}, \mathbf{i})$ can thus vary per pixel, comprising pixel-level sample counts $\mathbf{p}$ and image-level sample counts $\mathbf{i}$.

Optimization. The optimization itself is outlined in Alg. 1 (lines 1-5). We begin by generating samples using a pilot strategy $\mathbf{m}$, rendering an image $\mathbf{I} = \{\langle I_1\rangle, \ldots, \langle I_P\rangle\}$ and a second-moment image $\mathbf{M} = \{\langle M_1\rangle_1, \ldots, \langle M_P\rangle_N\}$ storing one moment for each candidate strategy for every pixel. These estimates are used to first decide all pixel-level sample counts and then, conditionally on those, the

image-level ones. For example, for VCM (Section 4) we first decide whether to enable merging for each pixel, then we determine the (global) number of light subpaths based on the per-pixel decisions.

Pixel-level optimization. The pixel-level sample counts $\mathbf{p}_j$ for a pixel j are optimized by finding the candidate strategy $\mathbf{n}$ with lowest work-normalized moment and selecting its per-pixel components:

$$(\mathbf{p}_j, \cdot) = \underset{\mathbf{n}\in\{\mathbf{n}^1 \dots \mathbf{n}^N\}}{\arg\min}\, M[\langle I_j \rangle_{\mathbf{n}}] C_j(\mathbf{n}), \tag{8}$$

where $C_j(\mathbf{n})$ comprises the per-pixel samples' cost plus $1/P$ of the per-image samples' cost. This minimization can be done via a simple linear search, as shown in Alg. 1 (lines 6-10). The robustness of the optimization is greatly increased by applying low-pass noise reduction on the second-moment image. It is also beneficial to apply a filter on the resulting sample-count image, since abrupt changes in sampling counts can cause visible artifacts. Note that this optimization does not necessarily minimize Eq. (7). Rather, it minimizes the sum of pixel efficiencies, which is a lower bound of Eq. (7). This simplification removes the need to pair-wise compare all candidates across all pixels, increasing performance but sacrificing the capability to balance sampling error across the image.

Image-level optimization. With the pixel-level counts fixed, the image-level counts can be optimized as sketched in Alg. 1 (lines 12-22). We find the sample counts $\mathbf{i}$ that minimize the relative work-normalized moment (7) given the pixel-level counts $\mathbf{p}_j$:

$$\mathbf{i} = \underset{\mathbf{i}\in\{\mathbf{i}^1 \dots \mathbf{i}^N\}}{\arg\min} \sum_{j=1}^{P} \frac{M[\langle I_j \rangle_{(\mathbf{p}_j,\mathbf{i})}]}{I_j^2} \sum_{j=1}^{P} C_j(\mathbf{p}_j, \mathbf{i}). \tag{9}$$

This is done by first accumulating the relative moments and costs over all pixels, and then again performing a simple linear search. Computing the relative moments requires accurate estimates of the pixel value I_j. We found that applying a denoiser on the image rendered with the pilot strategy (line 13) produces good results.

3.3 Estimating the moments

The key to efficient optimization is to cheaply compute the second moments of each candidate strategy $\mathbf{n} = (n_1, \dots, n_T)$ from a single run using the pilot strategy $\mathbf{m} = (m_1, \dots, m_T)$. To facilitate that, we rewrite the second moment (6) of each candidate in terms of the second moment of the pilot, with an additional per-sample correction factor. The second moment of the pilot can be trivially estimated by summing the square of its samples, and the correction factor allows us to efficiently compute the moments of all candidates on-the-fly from the same samples.

In Appendix B we show that the second moment (6) of a candidate strategy $\mathbf{n}$ can be written as

$$M[\langle I \rangle_{\mathbf{n}}] = \int_{\mathcal{X}} \frac{f^2(x) \sum_t c_t(x) w_{\mathbf{m},t}}{\sum_t c_t(x) m_t p_t(x)} \delta_{\mathbf{n},\mathbf{m}}(x)\, \mathrm{d}x, \tag{10}$$

where the fraction involves only the pilot strategy $\mathbf{m}$. The term

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \frac{\left(\sum_t \frac{m_t}{a_t} w_{\mathbf{a},t}(x)\right)^2 \sum_t c_t(x) \frac{n_t}{a_t} w_{\mathbf{a},t}(x)}{\left(\sum_t \frac{n_t}{a_t} w_{\mathbf{a},t}(x)\right)^2 \sum_t c_t(x) \frac{m_t}{a_t} w_{\mathbf{a},t}(x)} \tag{11}$$

Algorithm 1. Pseudo-code for our optimization. Given a pilot strategy $\mathbf{m}$, a set of candidate strategies $\mathbf{n}^1, \dots, \mathbf{n}^N$, and a cost function C, we render an image and estimate the second moments. Then, we first optimize all pixel-level sample counts $\mathbf{p}_j$, followed by all image-level sample counts $\mathbf{i}$.

```
1:  function FindBestStrategy(m, n^1, ..., n^N, C)
2:    M, I = ComputeMomentImage(m, n^1, ..., n^N)
3:    {p_j} = PixelLevelOptimize(M, n^1, ..., n^N, C)
4:    i = ImageLevelOptimize(M, I, n^1, ..., n^N, C, {p_j})
5:    return {p_j}, i   ← Set of pixel-level sample counts, image-level sample counts
6:  function PixelLevelOptimize(M, n^1, ..., n^N, C)
7:    {M̃_{1,1}, ..., M̃_{P,N}} = FilterImage({⟨M_1⟩_1 ... ⟨M_P⟩_N})
8:    for pixel index j = 1..P do                    ↖ Reduce noise in moments
9:      (p_j, _) = arg min M̃_{j,n} C_j(n)            ← Linear search
10:   return FilterImage({p_1, ..., p_P})
11:                                  ↖ Avoid abrupt changes in sample counts
12: function ImageLevelOptimize(M, I, n^1, ..., n^N, C, {p_j})
13:   {Ĩ_1, ..., Ĩ_P} = DenoiseImage(⟨I_1⟩, ..., ⟨I_P⟩)
14:   {R_i} = {0, ..., 0}      ← Init. total rel. moment per image-level candidate
15:   {C_i} = {0, ..., 0}      ← Initialize total cost per image-level candidate
16:   for pixel index j = 1..P do
17:     for all candidate strategy index k = 1..N do
18:       if p_j ≠ p^k then
19:         continue             ← Skip candidates with different local counts
20:       R_{i^k} += ⟨M_j⟩_k · Ĩ_j^{-2}          ← Accumulate relative moments
21:       C_{i^k} += C_j(n^k)                    ← Accumulate cost
22:   return arg min{R_i · C_i}   ← Pick best image-level counts via simple search
23: function ComputeMomentImage(m, n^1, ..., n^N)
24:   I = {⟨I_1⟩, ..., ⟨I_P⟩} = {0, ..., 0}      ← Initialize pixel estimates
25:   M = {⟨M_1⟩_1, ..., ⟨M_P⟩_N} = {0, ..., 0}  ← N moments per pixel
26:   a = {Σ_{k=1}^N n_k^1/N, ..., Σ_{k=1}^N n_k^T/N}   ← Set proxy strategy
27:   for pixel index j = 1..P do
28:     for technique index t = 1..T do
29:       for sample index i = 1..m_t do
30:         y = w_{m,t}(x_{t,i}) f_j(x_{t,i}) / (m_t p_t(x_{t,i}))   ← MIS-weighted pixel j estimate
31:         ⟨I_j⟩ += y      ↖ Precompute terms in δ_{n,m}(x_{t,i}) independent of n
32:         Precompute( (Σ_t m_t/a_t w_{a,t}(x_{t,i}))^2 / (Σ_t c_t(x_{t,i}) m_t/a_t w_{a,t}(x_{t,i})),
                        w_{a,1}(x_{t,i})/a_1, ..., w_{a,T}(x_{t,i})/a_T )
33:         for candidate strategy index k = 1..N do
34:           ⟨M_j⟩_k += y^2 · δ_{n^k,m}(x_{t,i})   ← Moment estimate (13)
35:   return M, I
```

is a correction factor involving both strategies, which for the classical balance heuristic (i.e., $c_t(x) = 1$) simplifies to

$$\delta^{\text{bal}}_{\mathbf{n},\mathbf{m}}(x) = \frac{\sum_t \frac{m_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathbf{a},t}(x)}. \tag{12}$$

The correction factor relies on a proxy strategy $\mathbf{a} = (a_1, \dots, a_T)$. While this strategy is theoretically unnecessary, it serves two practical purposes: (1) it enables computing the correction factor in a numerically stable way and (2) it separates out terms not involving n_t that can be precomputed and reused for all candidate moments. The proxy strategy can be defined arbitrarily, but for numerical stability it is beneficial for a_t and n_t to have similar orders. We simply set each a_t to the average of n_t across all candidate strategies.

The estimation of the second moment (10) for a given candidate strategy $\mathbf{n}$ is then relatively straightforward. Simply squaring the (MIS-weighted) contributions of the pilot samples yields an estimate for the integral without the correction factor $\delta_{\mathbf{n},\mathbf{m}}$. Thus, the only extra work required is to evaluate the correction for each sample (which is cheap thanks to the precomputed terms):

$$\langle M[\langle I\rangle_{\mathbf{n}}]\rangle_{\mathbf{m}} = \sum_t \sum_{i=1}^{m_t} \left(\frac{w_{\mathbf{m},t}(x_{t,i}) f(x_{t,i})}{m_t p_t(x_{t,i})} \right)^2 \delta_{\mathbf{n},\mathbf{m}}(x_{t,i}). \quad (13)$$

Lines 23-35 in Alg. 1 show how this moment computation can be integrated into a rendering iteration. For every sample, the proxy weights $w_{\mathbf{a},t}$ are computed once and shared among all correction-factor $\delta_{\mathbf{n},\mathbf{m}}$ evaluations. Appendix A shows how the same scheme can be used to efficiently estimate the first and second derivatives that arise in the convex optimization used by prior work.

4 APPLICATION AND RESULTS

Our main application is the vertex connection and merging (VCM) algorithm [Georgiev et al. 2012; Hachisuka et al. 2012]. It combines bidirectional path tracing (BDPT) [Veach and Guibas 1995a; Lafortune and Willems 1993] with photon mapping [Jensen 1996]. Paths are traced from the camera and the lights. Each ray along these paths might hit a light (or the camera), and next event estimation is done at every vertex to connect to a light (or the lens). On top of that, each vertex on a camera subpath is connected to some number of light-subpath vertices. Lastly, merging (a.k.a. photon density estimation) is performed at each camera-subpath vertex, except the first on the lens and the directly visible point.

Optimizing all parameters of VCM involves, e.g., setting the number of connections for each pixel, point in space, and camera-subpath length, which is prohibitively expensive. To keep the problem feasible, we reduce the degrees of freedom by only controlling

(1) n, the number of light paths traced,
(2) c, the number of connections as a global constant, and
(3) χ_j, whether to perform merging in each pixel j.

For the set of candidate strategies, we consider all possible combinations of (with P being the number of pixels)

$$n \in \left\{ \frac{1}{4}P, \frac{1}{2}P, \frac{3}{4}P, P, 2P \right\}, \quad c \in \{0, 1, 2, 4, 8, 16\}, \quad \chi_j \in \{0, 1\}, \quad (14)$$

as well as the special case ($n = 0, c = 0, \chi_j = 0\,\forall j$), i.e., unidirectional path tracing. Note that the number of light paths n is only allowed to be zero if all other bidirectional techniques are disabled.

4.1 Implementation

We have implemented our method on top of the public code of Grittmann et al. [2021]. All experiments were run on a 16-core AMD Ryzen 9 3950X processor with 64 GB of memory. The results shown in the following, unless stated otherwise, are equal-time renderings after 60s at a resolution of 640×480 (we also rendered some scenes for 25 min at 4K resolution, with similar results). We use the relative mean squared error (relMSE) as an error metric, which is an estimate of Eq. (4). Since the (rel)MSE is not robust to outliers, we ignore the 0.01% of pixels (i.e., 30 in total at our typical resolution) with highest error.

The optimization is done in up to two iterations, each rendering one sample per pixel. We start with unidirectional path tracing as the pilot strategy. If the outcome is to switch to a bidirectional technique, we optimize one more time with the samples of the bidirectional technique, which gives higher-quality second-moment estimates. If the pilot decides not to enable bidirectional sampling, no further optimization is done.

4.1.1 Cost heuristic. The cost of a strategy $\mathbf{n}$ is the expected time it takes to render one iteration with that strategy. We approximate that cost with a heuristic that roughly corresponds to the number of ray-tracing and shading operations:

$$C(n, c, \chi) = C_{\text{light}} \tilde{n}_{\text{l}} n + P \tilde{n}_{\text{c}} \left(C_{\text{cam}} + C_{\text{con}} c + C_{\text{m}} \tilde{n}_{\text{l}} n \tilde{n}_{\text{m}} \textstyle\sum_j \chi_j \right), \quad (15)$$

which is the sum of the cost (incl. next-event estimation) of tracing n light paths of average length $\tilde{n}_{\text{l}}$, and the cost of tracing P camera paths (i.e., one per pixel) of average length $\tilde{n}_{\text{c}}$ performing at each vertex next-event, connections, and merges. Here, $\tilde{n}_{\text{m}}$ is the average number of photons found by each density estimation, as a fraction of the total number of photons $\tilde{n}_{\text{l}} n$ in the scene.

The relative costs of the different techniques are controlled by four hyperparameters. C_{light} and C_{cam} are the combined costs of continuing the respective path at each vertex and performing next-event estimation. C_{con} is the cost of a single connection and C_{m} is the cost of a single merge operation. We determined the best values for our implementation by fitting the cost heuristic to brute-force measured render times across 25 test scenes: $C_{\text{cam}} = C_{\text{light}} = 1, C_{\text{con}} = 0.4$, and $C_{\text{m}} = 0.5$. These numbers are implementation-specific and likely differ between renderers. The average path lengths are determined on-the-fly from rendering statistics. When using a unidirectional pilot strategy, only $\tilde{n}_{\text{c}}$ is available, in which case we initialize the remaining statistics with an initial guess: $\tilde{n}_{\text{l}} = \tilde{n}_{\text{c}},\ \tilde{n}_{\text{m}} = 10^{-7}$.

The supplemental document shows that the above simple cost heuristic closely matches the actual render time in our tests. It also provides empirical evidence indicating that the accuracy of the cost heuristic is secondary compared to the other sources of error—using moments instead of variance and noise in the estimates.

4.1.2 Merge mask. Merging only benefits a specific type of effect, reflected or refracted caustics, that is often limited to small regions in the image. Hence, a global decision is apt to neglect these small regions to increase efficiency everywhere else. Therefore, we control the binary decision whether to perform merging on the pixel-level, by computing a *merge mask*.

Care has to be taken to avoid visible artifacts, since we only have a single sample per pixel. We apply a simple filtering scheme, shown in Fig. 2. First, the second-moment images are blurred (Gaussian filter, radius 8) to reduce noise and downsampled (averaged into 8×8-pixel tiles) to reduce overhead. Then, we run our pixel-level optimization. The resulting mask can still contain gaps from missing data, where merging in important regions may be incorrectly disabled. To combat that, we dilate the mask (box, radius 4). Finally, to remove discontinuities in the noise pattern from abruptly changing sample counts, which could cause visible artifacts, we blur the dilated mask (Gaussian, radius 4). The resulting mask after the blur contains floating point values between zero and one, which we take as the probability to perform merging at each vertex.

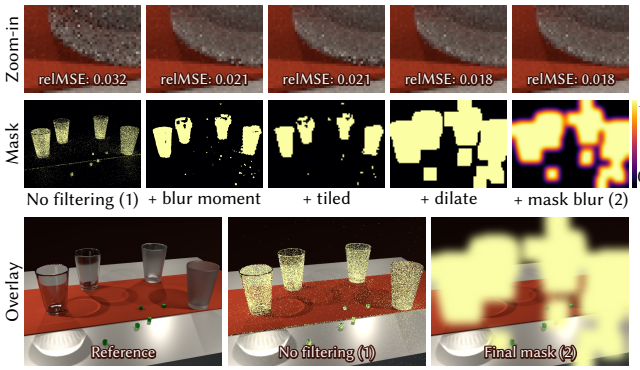

Fig. 2. When optimizing the pixel-level merging decisions χ_j, we apply a simple filtering scheme to increase robustness. The top row shows the effect of the different operations on the merge mask and the equal-time error. The bottom row shows an overlay of the mask over the reference image, without filtering and after applying our filtering.

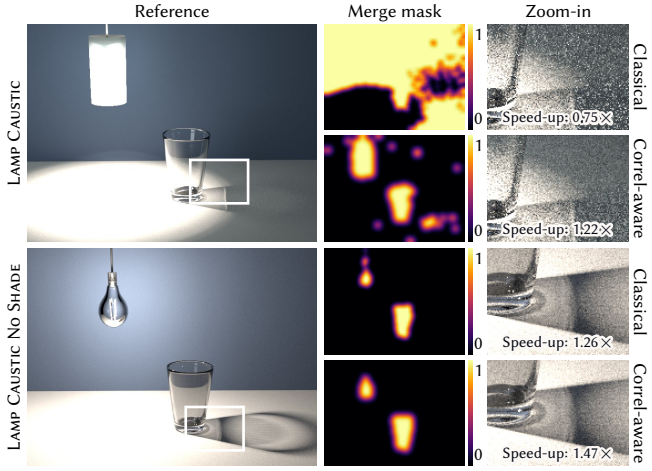

Fig. 3. A scene rendered with and without a lamp shade. We show the merge masks and the rendering speed-up due to our optimization when using the classical balance heuristic and Grittmann et al.'s [2021] correlation-aware weights. The lamp shade causes severe covariance in the merging techniques, and our optimization can further worsen the already poor performance of the classical balance heuristic. Using the correlation-aware weights avoids this problem by assigning low MIS weights to the problematic samples. Hence our optimization does not enable merges because they would not contribute to the combined estimate.

This simple filtering can be improved further with adaptive kernels and other ideas commonly used by denoising methods. The supplemental document discusses the impact of the different filters and their parameters.

4.1.3 Sample correlation. Sample correlation in the merging techniques can be a problem in VCM [Grittmann et al. 2021]. The second moments can grossly *underestimate* the actual variance of merging. Hence, optimizing the sample counts based on second moments (7) can produce suboptimal results. To circumnavigate this, we utilize

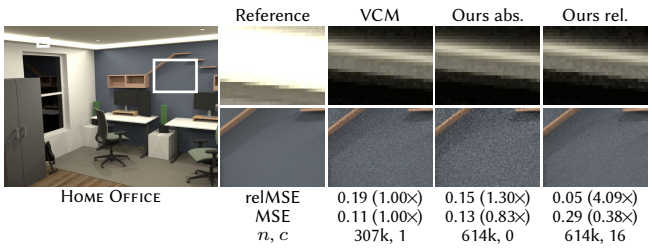

Fig. 4. In this scene, light tracing is the sole technique that can render the bright strip around the ceiling (top row, reduced exposure). Optimizing for absolute moments overfits on this bright region and disables all other techniques for the entire image. Using relative moments resolves the problem. Note that "Ours rel." yields lower relMSE but higher MSE than vanilla VCM.

the correlation-aware MIS weights of Grittmann et al. [2021]. The effect of optimizing the sample allocation w.r.t. these weights is shown in Fig. 3. It shows the merge masks and renderings with and without correlation-aware weights. We report speed-ups over vanilla VCM (i.e., $n = P$, $c = 1$, $\chi_j = 1$) using both the classical balance heuristic and the correlation-aware weights. Applying our moment-based optimization further amplifies the correlation problem when using the balance heuristic which already struggles with poor approximation from second moments. Applying it w.r.t. the correlation-aware weights produces consistent speed-ups. All results in the following use the correlation-aware weights for both the baseline and our method.

4.1.4 Relative error. We found that using the relative moments (7) (rather than absolute moments) is essential when optimizing the number of light paths n and global number of connections c. Figure 4 shows an example where the scene is dominated by indirect illumination from a bright strip of light along the ceiling. The direct illumination in the strip is best handled by light tracing. But all other illumination in the scene is best handled by bidirectional connections. Optimizing the absolute moments (i.e., omitting the pixel-value normalization) results in overfitting on the bright direct illumination and disables all connections. Optimizing with relative moments instead yields $c = 16$ connections and four times faster rendering. The caveat is that this approach requires good estimates/approximations of the ground-truth pixel values. To obtain those from a one-sample-per-pixel rendering, we denoise the image using Intel's Open Image Denoise [Áfra 2019].

4.1.5 Choosing the pilot strategy. We consider two options for the pilot strategy: forward path tracing (PT) or vanilla VCM ($n = P$, $c = 1$, $\chi_j = 1$). Figure 5 compares these options on two extreme cases, rendered for only 10s. In simple scenes that are best rendered unidirectionally, such as Modern Living Room, using vanilla VCM as the pilot reduces performance for shorter render times, due to the wasted bidirectional samples in the first iteration. With PT as the pilot, the overhead is limited to that of the optimization. For scenes like Sponge, which is illuminated solely by a caustic, starting with VCM provides the best performance because there the unidirectional samples from a PT pilot are wasted. However, these unidirectional

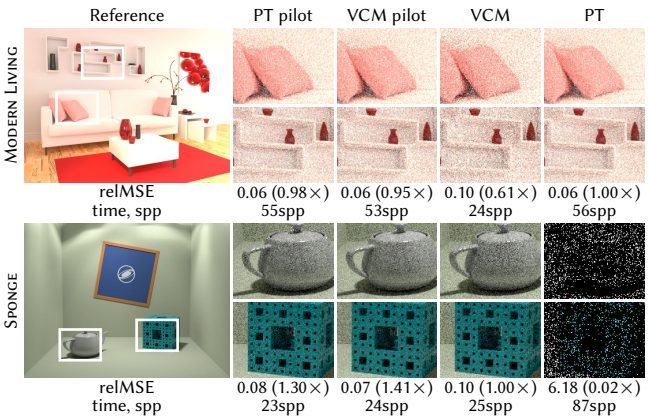

Fig. 5. Impact of the pilot strategy in short renderings. We show the relMSE after 10s (lower is better) and the speed-up in parentheses compared to the baseline (higher is better). Modern Living Room does not benefit from bidirectional methods. Using forward path tracing (PT) as the pilot limits the overhead to that of the optimizer (2%). Starting with VCM incurs additional overhead from (wasted) bidirectional samples. The solely caustic illumination in the Sponge scene is difficult to render with PT. Nevertheless, using PT as a pilot gives enough information to update the sampling strategy and still achieve faster rendering than vanilla VCM.

samples are much cheaper than bidirectional ones, and they provide sufficient information for our optimization.

An advantage of a vanilla VCM pilot is the lower noise in the second moment estimates used by our optimization. We found that PT and VCM pilots produce very similar results when optimizing image-level parameters, e.g., number of light paths n and number of connections c. However, for pixel-level optimization, a single sample per pixel from a PT pilot is insufficient.

Our solution is to optimize in two stages. We start with a PT pilot and only image-level optimization. If that optimization enables bidirectional sampling, we render one iteration with the optimized bidirectional strategy. The second moments estimated from that iteration are then used to perform a full optimization, including the pixel-level merge mask. This hybrid pilot minimizes overhead in simple scenes and produces the same optimization as a VCM pilot in difficult scenes.

The rendered image of the pilot can be averaged into the final result. However, the PT pilot may have excessive noise that will not vanish quickly. Thus, if bidirectional sampling is enabled by the PT pilot, we discard both the rendered image and the second moments from the PT pilot, and start from scratch with our optimized VCM pilot. The rendered image of the VCM pilot is averaged with the subsequent iterations, as it has a similar level of noise.

4.2 Results

We tested our method on bidirectional path tracing (BDPT) and full VCM. An overview of the results across our 25 test scenes (22 for BDPT) is shown in Table 1. Our optimized BDPT achieves 18% faster rendering on average than vanilla BDPT ($n = P, c = 1$). Our optimized VCM achieves consistent speed-ups of up to 5× over vanilla VCM ($n = P, c = 1, \chi_j = 1$) across all scenes; the worst case

Table 1. Statistics of the speed-up (higher is better) of our method across the 25 test scenes of the VCM application and the 22 scenes of the BDPT application. Computed after 60s rendering, averaged across 5 runs, using the relMSE error metric with outlier removal.

Speed-up	vs. path tracing		vs. vanilla	
	BDPT	VCM	BDPT	VCM
Average	3.14×	5.00×	1.18×	1.68×
Worst	0.99×	0.98×	0.95×	1.12×
Best	97.09×	600.77×	1.60×	5.32×

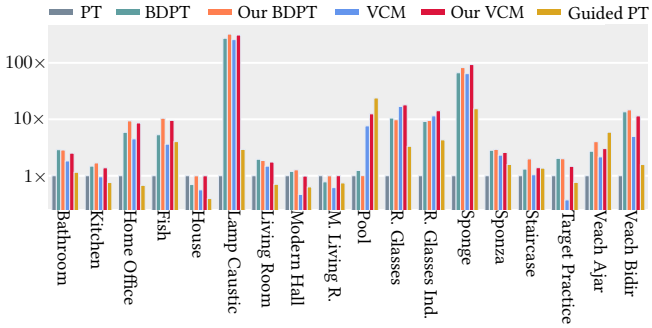

Fig. 6. Speed-up in terms of relMSE of different methods over unidirectional path tracing. 'Guided PT' is the path guiding method of Ruppert et al. [2020]. Our method consistently outperforms unidirectional PT and vanilla VCM.

is still 12% faster. This is because merging is expensive and often not beneficial. In both variants, our method performs consistently faster than unidirectional path tracing (PT), with a worst-case slowdown of 2% and a best-case speed-up of 600×.

Figure 6 plots the speed-ups over PT of all methods for multiple test scenes. It also compares the performance to that of guided forward path tracing [Ruppert et al. 2020]. The supplemental materials provide an interactive viewer with all rendered images.

4.2.1 Bidirectional path tracing. Figure 7 shows two scenes rendered with forward path tracing, vanilla BDPT, and our adaptive BDPT. Home Office is dominated by diffuse indirect illumination which benefits from many bidirectional connections. Modern Hall is overall well-handled by forward path tracing. By reducing the number of light paths, our method finds a sweet-spot providing a minor increase in performance. In both scenes, the overhead due to the unidirectional pilot iteration reduces performance initially, but for longer renderings our method performs consistently better than both baselines.

The first two columns of plots in Fig. 7 compare our estimated work-normalized second moments to ground-truth work-normalized variances. The ground-truth is obtained by brute-force rendering with different sample counts and computing the product of relMSE and render time. Error values are plotted for different n given our decision for c (first column) and for different c, given our decision for n (second column). Our estimates do not perfectly match the ground truth, though they yield the same or very similar minima, which suffices for our optimization.

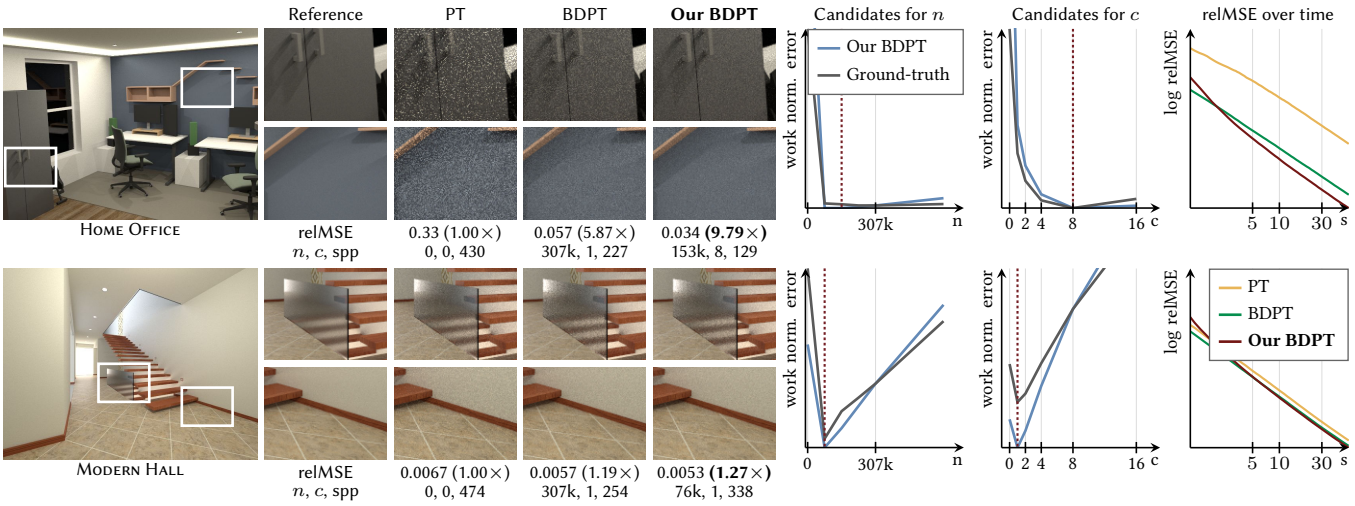

Fig. 7. Equal-time (60s) comparison between our optimized BDPT and two baselines. n and c are the number of light subpaths and bidirectional connections, respectively. The numbers in parentheses are the speed-ups (higher is better) over forward path tracing (PT). The plots compare our estimated work-normalized moments to ground truth work-normalized variances for different choices of n and c. The dashed red line marks the sample count chosen by our optimization. Basing our optimization on the second moments produces similar sample counts as the much more expensive full variances and yields consistent equal-time speed-ups compared to both baselines.

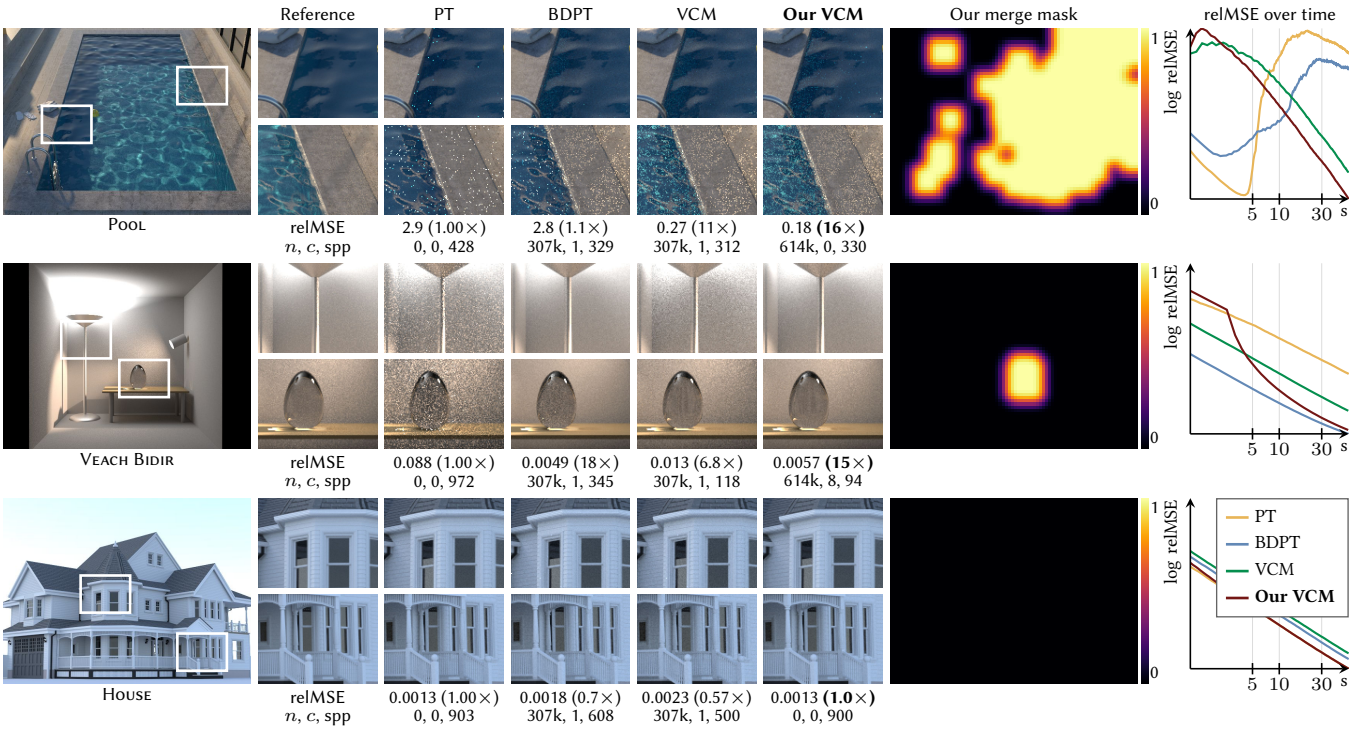

Fig. 8. Equal-time (60s) comparison between our optimized VCM and three baselines. The numbers in parentheses are the speed-up (higher is better) over forward path tracing (PT). The false-color image visualizes our per-pixel decision whether to perform merging. The Pool scene is dominated by caustics and benefits mostly from merging and light tracing. Veach Bidir is mostly indirectly illuminated and features a small glass egg, benefiting from bidirectional connections and local merging. House is best rendered unidirectionally, and our method sticks to PT, incurring less than 1% overhead due to the optimization.

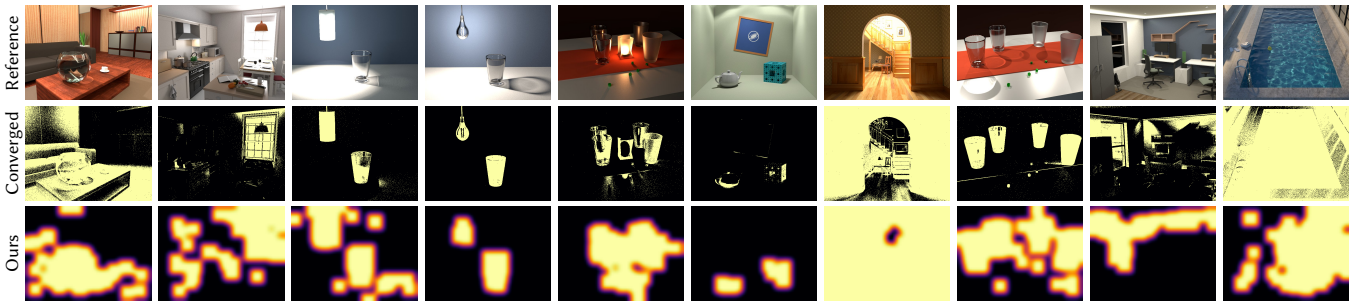

Fig. 9. Comparison of our aggressively filtered merge masks against ground-truth masks. Our masks are computed from 1-spp VCM with image-level optimization. The converged masks are based on second-moment estimates obtained from 4096 iterations of vanilla VCM.

4.2.2 Full VCM. Adding merging into the mix of techniques enables robust rendering of reflected and refracted caustics. In Fig. 8 we compare the performance of our optimized VCM to different baselines. In both VCM variants we use correlation-aware MIS weighting [Grittmann et al. 2021].

The exterior, environment-map lit Pool scene contains a caustic that is very challenging for unidirectional path tracing to sample. VCM performs much better but also struggles with the very low photon density. Our optimization improves efficiency by using the maximum allowed number of light paths, disabling connections, and limiting merges to the region containing refracted caustics.

The Veach Bidir scene [Veach and Guibas 1995a] was originally modelled to showcase BDPT. Naturally, it benefits from many light paths and connections, but also features a caustic that benefits from merging. The merge mask generated by our method restricts the costly merges to the caustic image region. The error over time (last column) in this scene reveals the impact of starting with a PT pilot. This first unidirectional iteration is wasted. Hence, our method performs worse than vanilla VCM for short renderings of less than three seconds. Note that our optimized VCM has a numerically higher error than vanilla BDPT, but produces a qualitatively better image. The reason is that the reflected caustic in the glass egg manifests in BDPT as two outlier pixels that are ignored by our error metric. Our VCM captures this caustic almost perfectly, at the cost of slightly increased error elsewhere.

Lastly, the diffuse exterior House scene is a case where bidirectional sampling is wasteful. Vanilla VCM and BDPT are much slower than forward path tracing in this case. Our optimized VCM completely avoids tracing paths from the lights and only incurs a tiny ($< 1\%$) overhead from the optimization. The result is visually indistinguishable from the PT image.

4.2.3 Merge masks. Obtaining a reliable merge mask from a few (or even one) samples per pixel requires filtering that noisy data. Figure 9 compares our masks to ones obtained from ground-truth second moments computed from 4096 iterations of vanilla VCM. Our filtered 1-spp mask covers all crucial regions but can, of course, be improved further. An interesting insight is that merges are mostly (though not always) useful in pixels that see highly glossy surfaces. A heuristic that limits merging to such pixels, based directly on material properties, could eliminate the overhead of our merge-mask computation. Such a heuristic would be reflected in the MIS weights, as part of the effective density $np(x)$, hence it trivially integrates into our optimization.

4.2.4 Overhead. The main computational cost in our approach is the large number of moments that are estimated and processed. The VCM application computes 61 second moments per pixel (one for each candidate from Eq. (14)), which requires ~500 MB of memory at full-HD image resolution. This cost can be reduced by computing per-tile instead of per-pixel moments (an 8×8 tiling reduces the full-HD memory consumption to ~8 MB).

The overhead of our method also depends on the outcome of the optimization after the initial path-tracing (PT) pilot run. If the decision is to stick to PT, the overhead of our implementation is on average 82% (~141 ms) of the cost of tracing one path per pixel. This comprises the cost of denoising the image (the main bottleneck) and accumulating and processing the per-pixel moments for the image-level decisions.

If the decision after the PT pilot is to discard the rendering and switch to VCM, the overhead increases by the cost of the discarded PT iteration and the cost of constructing the merge mask. In our VCM implementation it amounts on average to 3.3× (~1.3 s) the cost of a single iteration of our optimized VCM. Most of that overhead is due to the filtering applied when constructing the merge mask. Hence, in our BDPT implementation, the overhead is only 1.5× (~498ms) the cost of a single iteration of our optimized BDPT.

Note that we have not spent much effort on optimizing our code. The overhead can potentially be reduced significantly: Our optimization consists solely of trivially parallelizable image processing operations that could, e.g., be run on a GPU. We leave such engineering to future work, since the results are already consistently faster than unidirectional path tracing, despite the overhead.

5 DISCUSSION AND FUTURE WORK

The three main limitations of our method are the overhead, error due to noisy estimates, and error due to approximations. Also, the quality of the results and the range of possible applications can be further improved by making the decisions more local, accounting for the effect of changing sampling densities (e.g., guiding), or even jointly optimizing the MIS weights [Kondapaneni et al. 2019], sampling densities [Karlík et al. 2019], and sample allocations.

Divide-and-conquer optimization. The overhead of our optimization can be reduced by reducing the number of candidate sample-allocation strategies. Additionally, the set of candidates could be altered between iterations. For example, a divide and conquer approach could start with just two candidates in the first iteration. The second iteration would then generate new candidates, e.g. by perturbing the better of the initial two. Iteratively refining the decision in this way drastically reduces the overhead, but it can also take longer to find the best strategy.

Noisy estimates. Optimizing the per-pixel merging decisions requires careful filtering, since the optimization is based on the samples of a single iteration. The estimation noise has been the biggest problem in our evaluation, with the filtering having a significant impact on the result quality. Very noisy input samples can lead to poor pixel-level decisions, which in turn could produce visible artifacts. This problem is similar to that in adaptive sampling methods which decide on the number of samples per pixel based on variance estimates [Zwicker et al. 2015]. Future work could make our method more robust by transferring advances from adaptive sampling and reconstruction to our context. An interesting idea would be to replace our simple filtering pipeline by a learning approach, i.e., training a specialized denoising network to construct a merge mask. A simpler option is to use multiple pilot iterations to estimate the second moments before optimizing the sample counts, potentially driving that process by adaptive sampling too.

Approximation error. A key reason why our brute-force scheme works well in practice is that the full variance is approximated by the second moments. However, that is also the main source of approximation error, as we detail in the supplemental document. Only considering the second moments can lead to suboptimal sample allocations, e.g., in the presence of sample correlation. More accurate approximations, or even unbiased estimates of the full variance are apt to improve the results further. Also, we have ignored the fact that vertex merging (i.e., photon mapping) is biased. Estimating that bias and incorporating it into our objective, ideally without having to perform merging, may also improve results.

Optimization granularity. Our sample-allocation optimization is carried out in screen space, but it is possible to consider a finer granularity. For example, the number of bidirectional connections could be optimized within spatial regions in a scene. While increasing the accuracy, making decisions more localized also hazards the robustness, as fewer samples are available to compute the required quantities.

Unknown sampling densities. Optimizing the MIS sample allocation requires knowledge of the sampling densities of the different techniques. However, these densities might themselves be subject to change, or completely unknown. For example, photon emission guiding [Vorba et al. 2014; Grittmann et al. 2018] produces densities that change over time, while Markov chain Monte Carlo approaches [Veach and Guibas 1997; Šik and Křivánek 2018; Šik and Křivánek 2019; Šik et al. 2016] are unable to compute the exact densities in the first place. Future work could look into approximations that predict the densities of such techniques. For example, by pretending that they are proportional to the target function [Kelemen et al. 2002].

6 CONCLUSION

We propose a method to automatically adapt the set of sampling techniques in MIS, and their sample counts, to a given input. Our application focuses on bidirectional rendering algorithms, but our method is general and applicable to any MIS combination. The key ingredient is a numerically robust and computationally efficient scheme for estimating the second moments of different sample-allocation strategies.

In practice, our adaptive VCM implementation never performs significantly worse than plain unidirectional path tracing, even on simple scenes. At the same time, complex scenes with strong indirect lighting and reflected caustics are rendered more efficiently than vanilla VCM with fixed parameters.

Rendering efficiency often relies on manual per-scene parameter tuning. The consistent speed-ups achieved by our method show that it is possible, and beneficial, to automate this tedious process.

ACKNOWLEDGMENTS

The figures were made with the help of Mira Niemann's figure generator. We thank the following people for sharing the test scenes and assets: Wig42 (Modern Living Room, Staircase), Ondřej Karlík (Pool), Jay-Artist (Kitchen), oldtimer and ColeHarris (Fish), and Vladislav Hnatovskiy and Mira Niemann (Target Practice). This work was supported by Velux Stiftung project 1350. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956585 (https://prime-itn.eu).

REFERENCES

Attila T. Áfra. 2019. Intel® Open Image Denoise. https://www.openimagedenoise.org/.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6 (2012), 10 pages.

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

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH Asia 2019)* 38, 6 (Nov. 2019), 9 pages.

Pascal Grittmann, Arsène Pérard-Gayot, Philipp Slusallek, and Jaroslav Křivánek. 2018. Efficient Caustic Rendering with Lightweight Photon Mapping. *Comput. Graph. Forum (EGSR '18)* 37, 4 (2018), 133–142.

Pascal Grittmann, Ömercan Yazici, Iliyan Georgiev, and Philipp Slusallek. 2022. *Implementation of Efficiency-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms.* https://doi.org/10.5281/zenodo.6514204

Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed metropolis light transport. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph. (TOG)* 31, 6 (2012), 10 pages.

Vlastimil Havran and Mateu Sbert. 2014. Optimal Combination of Techniques in Multiple Importance Sampling. In *Proc. VRCAI '14* (Shenzhen, China). ACM, New York, NY, 141–150.

Hera Y. He and Art B. Owen. 2014. Optimal mixture weights in multiple importance sampling. https://doi.org/10.48550/ARXIV.1411.3954

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. In *Eurographics Workshop on Rendering Techniques*. Springer, 326–335.

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Eurographics workshop on Rendering techniques*. Springer, 21–30.

Ondřej Karlík, Martin Šik, Petr Vévoda, Tomáš Skřivan, and Jaroslav Křivánek. 2019. MIS Compensation: Optimizing Sampling Techniques in Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH Asia '19)* 38, 6 (2019), 12 pages.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. *Comput. Graph. Forum* 21, 3 (2002), 531–540.

Ivo Kondapaneni, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Trans. Graph. (SIGGRAPH 2019)* 38, 4 (July 2019), 14 pages.

Jaroslav Křivánek, Christophe Chevallier, Vladimir Koylazov, Ondřej Karlíik, Henrik Wann Jensen, and Thomas Ludwig. 2018. Realistic Rendering in Architecture and Product Visualization. In *ACM SIGGRAPH 2018 Courses*. Article 10, 5 pages. https://doi.org/10.1145/3214834.3214872

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. 93 (Dec. 1993), 145–153.

Heqi Lu, Romain Pacanowski, and Xavier Granier. 2013. Second-Order Approximation for Variance Reduction in Multiple Importance Sampling. *Comput. Graph. Forum* 32, 7 (2013), 131–136.

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. https://doi.org/10.1145/3305366.3328091

David Murray, Sofiane Benzait, Romain Pacanowski, and Xavier Granier. 2020. On Learning the Best Balancing Strategy. In *Eurographics 2020*, Vol. 20. 1–4.

Kosuke Nabata, Kei Iwasaki, and Yoshinori Dobashi. 2020. Resampling-aware Weighting Functions for Bidirectional Path Tracing Using Multiple Light Sub-Paths. *ACM Trans. Graph. (SIGGRAPH 2020)* 39, 2 (2020), 1–11.

Art Owen and Yi Zhou. 2000. Safe and Effective Importance Sampling. *J. Amer. Statist. Assoc.* 95, 449 (2000), 135–143.

Anthony Pajot, Loic Barthe, Mathias Paulin, and Pierre Poulin. 2010. Representativity for robust and adaptive multiple importance sampling. *IEEE transactions on visualization and computer graphics* 17, 8 (2010), 1108–1121.

Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. 2015. Probabilistic connections for bidirectional path tracing. *Comput. Graph. Forum* 34, 4 (2015), 75–86.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-Aware Path Guiding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020)* 39, 4 (July 2020), 12 pages.

Lukas Ruppert, Sebastian Herholz, and Hendrik PA Lensch. 2020. Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 15 pages.

Mateu Sbert and Víctor Elvira. 2022. Generalizing the balance heuristic estimator in multiple importance sampling. *Entropy* 24, 2 (2022), 191.

Mateu Sbert and Vlastimil Havran. 2017. Adaptive multiple importance sampling for general functions. *The Visual Computer* 33, 6 (2017), 845–855.

Mateu Sbert, Vlastimil Havran, and László Szirmay-Kalos. 2016. Variance Analysis of Multi-sample and One-sample Multiple Importance Sampling. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 451–460.

Mateu Sbert, Vlastimil Havran, and Laszlo Szirmay-Kalos. 2018a. Multiple importance sampling revisited: breaking the bounds. *EURASIP Journal on Advances in Signal Processing* 2018, 1 (2018), 1–15.

Mateu Sbert, Vlastimil Havran, and László Szirmay-Kalos. 2019. Optimal Deterministic Mixture Sampling.. In *Eurographics (Short Papers)*. 73–76.

Mateu Sbert, Vlastimil Havran, László Szirmay-Kalos, and Víctor Elvira. 2018b. Multiple importance sampling characterization by weighted mean invariance. *The Visual Computer* 34, 6 (2018), 843–852.

Martin Šik and Jaroslav Křivánek. 2018. Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation. *IEEE Transactions on Visualization and Computer Graphics* 26, 4 (2018), 1821–1840.

Martin Šik and Jaroslav Křivánek. 2019. Implementing One-Click Caustics in Corona Renderer. (2019).

Martin Šik, Hisanari Otsu, Toshiya Hachisuka, and Jaroslav Křivánek. 2016. Robust light transport simulation via metropolised bidirectional estimators. *ACM Trans. Graph. (TOG)* 35, 6 (2016), 245.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation.* Stanford University PhD thesis.

Eric Veach and Leonidas Guibas. 1995a. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.

Eric Veach and Leonidas J Guibas. 1995b. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *SIGGRAPH '95*. ACM, 419–428.

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 65–76.

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2014)* 33, 4 (2014), 11 pages.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Comput. Graph. Forum* 34, 2 (2015), 667–681.

A CONVEX OPTIMIZATION

The objective in Eq. (7) can be made convex under two simplifying assumptions: if the classical balance heuristic is used, i.e., $c_t(x) = 1 \forall t$, and the cost is linear in the sample counts, i.e., $C(\mathbf{n}) = \sum_t \gamma_t n_t$. In that case,

$$\sum_i \frac{M[\langle I \rangle_\mathbf{n}]}{I_i^2} C(\mathbf{n}) = \sum_i \frac{\sum_t \gamma_t n_t}{I_i^2} \int_\mathcal{X} \frac{f^2(x)}{\sum_t n_t p_t(x)} \,\mathrm{d}x \tag{16a}$$

$$= \sum_i \frac{1}{I_i^2} \int_\mathcal{X} \frac{f^2(x)}{\sum_t r_t \frac{p_t(x)}{\gamma_t}} \,\mathrm{d}x =: W(\mathbf{r}), \tag{16b}$$

where

$$r_t = \frac{\gamma_t n_t}{\sum_k \gamma_k n_k} \in [0, 1] \tag{17}$$

is the relative computation time invested into technique t. Our task thus reduces to finding the optimal $\mathbf{r} = (r_1 \dots \mathbf{r}_T)$. Since the additional γ_t factors are constant, $W(\mathbf{r})$ in Eq. (16b) is a convex function in the cost ratios r_t, because its partial derivatives have the same form as those used by previous work [Sbert et al. 2019; Murray et al. 2020] and satisfy the same conditions (i.e., yield a positive-definite Hessian).

Hence, we can apply the same Newton-Raphson root-finding as previous work and find the optimal sampling ratio vector $\mathbf{r}$ by iteratively updating

$$\mathbf{r}_n = \mathbf{r}_{n-1} - \mathcal{H}_\mathbf{r}^{-1}[W]\,(\mathbf{r}_{n-1})\ \nabla_\mathbf{r} W\,(\mathbf{r}_{n-1})\,, \tag{18}$$

starting with an initial guess r_0. Each element of the Hessian $\mathcal{H}_\mathbf{r}[W]$ and gradient $\nabla_\mathbf{r} W$ is given by a sum of integrals identical to those of previous work [Murray et al. 2020], except that each PDF $p_t(x)$ is divided by the sample cost γ_t. For example,

$$\frac{d}{dr_t} W = \sum_i \frac{1}{I_i^2} \int_\mathcal{X} \frac{f^2(x) \left(\frac{p_{T-1}(x)}{\gamma_{T-1}} - \frac{p_t(x)}{\gamma_t} \right)}{\left(\sum_{t'} r_{t'} \frac{p_{t'}(x)}{\gamma_{t'}} \right)^2} \,\mathrm{d}x, \tag{19}$$

where, like previous work, we have used the fact that the ratios must sum to one, to replace the last ratio $r_T = 1 - \sum_{t<T-1} r_t$ by one minus the sum of all others.

The partial derivatives (19) suffer from a similar problem regarding numerical robustness as the second moments: The numerator and denominator inside the integral contain terms involving the PDFs. This is fine for simple applications like direct illumination, where the low-dimensional PDFs can be represented by floating point arithmetic. But, e.g., for bidirectional algorithms, naively computing the PDFs directly is not an option.

Following the same idea as for our second moment estimation, we re-write the derivatives based solely on MIS weights. Here, we

discuss the example of the first derivative (19), but the second derivatives can be computed in exactly the same manner:

$$\int_{\mathcal{X}} \frac{f^2(x)\left(\frac{p_{T-1}(x)}{\gamma_{T-1}} - \frac{p_t(x)}{\gamma_t}\right)}{\left(\sum_{t'} r_{t'} \frac{p_{t'}(x)}{\gamma_{t'}}\right)^2} \mathrm{d}x \tag{20a}$$

$$= C(\mathbf{n})^2 \int_{\mathcal{X}} \left[\frac{f^2(x)\frac{p_{T-1}(x)}{\gamma_{T-1}}}{\left(\sum_{t'} n_{t'} p_{t'}(x)\right)^2} - \frac{f^2(x)\frac{p_t(x)}{\gamma_t}}{\left(\sum_{t'} n_{t'} p_{t'}(x)\right)^2} \right] \mathrm{d}x \tag{20b}$$

$$= C(\mathbf{n})^2 \int_{\mathcal{X}} \left[\frac{f^2(x)\frac{w_{\mathbf{n},T-1}(x)}{\gamma_{T-1}\, n_{T-1}}}{\sum_{t'} n_{t'} p_{t'}(x)} - \frac{f^2(x)\frac{w_{\mathbf{n},t}(x)}{\gamma_t\, n_t}}{\sum_{t'} n_{t'} p_{t'}(x)} \right] \mathrm{d}x \tag{20c}$$

That is, the derivatives can be computed exactly like the second moments, because they essentially *are* second moments—just with some extra MIS weight(s) multiplied on them. Thus, we can use the same steps as Eq. (24a) and Eq. (25) to convert an arbitrary pilot moment into the desired derivatives.

This extends the previous convex optimization approaches by additionally taking sample cost into account, and averaging decisions over an image. It also enables optimization without using the initial guess, or intermediate results, as actual sampling strategies.

B ROBUST MOMENT ESTIMATION

We are interested in estimating the second moment $M[\langle I\rangle_{\mathbf{n}}]$ of a candidate strategy $\mathbf{n}$ using the samples of another strategy $\mathbf{m}$. Simply squaring and summing up the contributions of those samples yields an unbiased estimate of the second moment of $\mathbf{m}$:

$$\sum_t \sum_{i=1}^{m_t} \left(\frac{w_{\mathbf{m},t}(x_{t,i}) f(x_{t,i})}{m_t p_t(x_{t,i})} \right)^2 \approx \int_{\mathcal{X}} \frac{f^2(x) \sum_t c_t(x) w_{\mathbf{m},t}(x)}{\sum_t c_t(x) m_t p_t(x)} \mathrm{d}x = M[\langle I\rangle_{\mathbf{m}}].$$

To arrive at an estimator for the desired second moment $M[\langle I\rangle_{\mathbf{n}}]$, we write $M[\langle I\rangle_{\mathbf{n}}]$ in terms of the above integral but with an additional correction factor:

$$M[\langle I\rangle_{\mathbf{n}}] = \int_{\mathcal{X}} \frac{f^2(x) \sum_t c_t(x) w_{\mathbf{m},t}(x)}{\sum_t c_t(x) m_t p_t(x)} \delta_{\mathbf{n},\mathbf{m}}(x)\, \mathrm{d}x, \tag{21}$$

where

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \frac{\sum_t c_t(x) m_t p_t(x)}{\sum_t c_t(x) n_t p_t(x)} \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)} \tag{22}$$

simply replaces the sums in the numerator and the denominator with the desired ones. To obtain the desired estimator, we only need to additionally multiply each squared sample contribution by $\delta_{\mathbf{n},\mathbf{m}}(x_{t,i})$, as we do in Eq. (13). Unfortunately, severe loss of numerical precision can be incurred when computing the involved sums of PDFs whose magnitudes can vary wildly. To that end, we can rewrite these sums in terms of MIS weights which can be evaluated in a numerically robust manner:

$$\frac{\sum_t c_t(x) m_t p_t(x)}{\sum_t c_t(x) n_t p_t(x)} = \sum_t \frac{m_t}{n_t} \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} = \sum_t \frac{m_t}{n_t} w_{\mathbf{n},t}(x).$$

Substituting this result into Eq. (22) yields a numerically robust expression for the correction factor:

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \left(\sum_t \frac{m_t}{n_t} w_{\mathbf{n},t}(x) \right) \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)}. \tag{23}$$

However, the evaluation of this expression can be inefficient: It would require computing (and storing) the weights $w_{\mathbf{n},t}$ and $w_{\mathbf{m},t}$ for every sample $x_{t,i}$, technique t, and candidate strategy $\mathbf{n}$. For complex applications like bidirectional path tracing, this will quickly become expensive as each weight computation requires a full sweep over the vertices of the path represented by $x_{t,i}$. To that end, we express the MIS weights of any strategy in terms of the weights of hypothetical strategy $\mathbf{a} = (a_1, \ldots, a_T)$, with $a_t > 1\,\forall t$, using similar manipulations as above:

$$w_{\mathbf{n},t}(x) = \frac{c_t(x) n_t p_t(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{24a}$$

$$= \frac{n_t}{a_t} \frac{c_t(x) a_t p_t(x)}{\sum_k c_k(x) a_k p_k(x)} \frac{\sum_{t'} c_{t'}(x) a_{t'} p_{t'}(x)}{\sum_{t'} c_{t'}(x) n_{t'} p_{t'}(x)} \tag{24b}$$

$$= \frac{\frac{n_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathbf{a},t'}(x)}. \tag{24c}$$

We substitute this result three times into Eq. (23) to obtain

$$\delta_{\mathbf{n},\mathbf{m}}(x) = \left(\sum_t \frac{m_t}{n_t} \frac{\frac{n_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_{t'} \frac{n_{t'}}{a_{t'}} w_{\mathbf{a},t'}(x)} \right) \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{\sum_t c_t(x) w_{\mathbf{m},t}(x)} \tag{25}$$

$$= \frac{\sum_t \frac{m_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathbf{a},t}(x)} \frac{\sum_t c_t(x) w_{\mathbf{n},t}(x)}{1} \frac{1}{\sum_t c_t(x) w_{\mathbf{m},t}(x)}$$

$$= \frac{\sum_t \frac{m_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathbf{a},t}(x)} \frac{\sum_t c_t(x) \frac{n_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_t \frac{n_t}{a_t} w_{\mathbf{a},t}(x)} \frac{\sum_t \frac{m_t}{a_t} w_{\mathbf{a},t}(x)}{\sum_t c_t(x) \frac{m_t}{a_t} w_{\mathbf{a},t}(x)},$$

which is identical to the expression in Eq. (11).