Selectively Metropolised Monte Carlo light transport simulation

BENEDIKT BITTERLI and WOJCIECH JAROSZ, Dartmouth College, USA

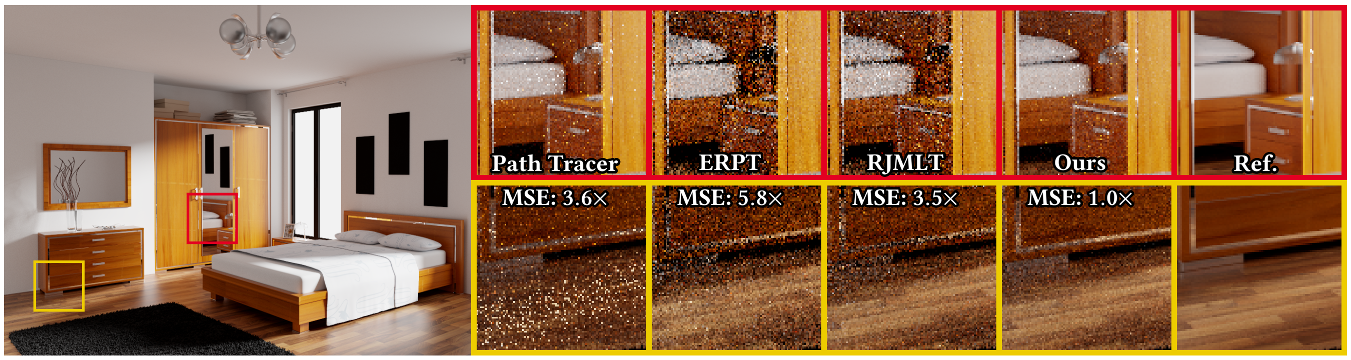

Fig. 1. For this indoor scene lit by a sun/sky environment emitter, path tracing resolves most of the lighting well, but introduces high variance in the reflective caustic cast by the mirror. ERPT and RJMLT resolve the caustic better, but do not handle diffuse lighting well and introduce noise and temporal flickering. Our method uses MLT to only resolve the difficult paths using bidirectional path tracing, and handles everything else with path tracing, leading to significantly improved MSE (relative MSE shown in insets) and vastly reduced temporal flickering (see Fig. 5). Scene ©SlykDrako.

Light transport is a complex problem with many solutions. Practitioners are now faced with the difficult task of choosing which rendering algorithm to use for any given scene. Simple Monte Carlo methods, such as path tracing, work well for the majority of lighting scenarios, but introduce excessive variance when they encounter transport they cannot sample (such as caustics). More sophisticated rendering algorithms, such as bidirectional path tracing, handle a larger class of light transport robustly, but have a high computational overhead that makes them inefficient for scenes that are not dominated by difficult transport. The underlying problem is that rendering algorithms can only be executed indiscriminately on all transport, even though they may only offer improvement for a subset of paths. In this paper, we introduce a new scheme for selectively combining different Monte Carlo rendering algorithms. We use a simple transport method (e.g. path tracing) as the base, and treat high variance "fireflies" as seeds for a Markov chain that locally uses a Metropolised version of a more sophisticated transport method for exploration, removing the firefly in an unbiased manner. We use a weighting scheme inspired by multiple importance sampling to partition the integrand into regions the base method can sample well and those it cannot, and only use Metropolis for the latter. This constrains the Markov chain to paths where it offers improvement, and keeps it away from regions already handled well by the base estimator. Combined with stratified initialization, short chain lengths and careful allocation of samples, this vastly reduces non-uniform noise and temporal flickering artifacts normally encountered with a global application of Metropolis methods. Through careful design choices, we ensure our algorithm never performs much worse than the base estimator alone, and usually performs significantly better, thereby reducing the need to experiment with different algorithms for each scene.

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

Authors' address: Benedikt Bitterli, benedikt.m.bitterli.gr@dartmouth.edu; Wojciech Jarosz, wojciech.k.jarosz@dartmouth.edu, Dartmouth College, Department of Computer Science, 9 Maynard St. Hanover, NH, 03755, USA.

Additional Key Words and Phrases: Ray tracing, photorealistic rendering

ACM Reference Format:
Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively Metropolised Monte Carlo light transport simulation. *ACM Trans. Graph.* 38, 6, Article 153 (November 2019), 10 pages. https://doi.org/10.1145/3355089.3356578

1 INTRODUCTION

Photorealistic image synthesis is an important problem with many applications in scientific visualization, video games and the movie industry. The rendering equation [Immel et al. 1986; Kajiya 1986] and the path integral [Veach 1997] form the predominant mathematical framework for physically based rendering, formalizing it in terms of an integration problem over the space of light paths connecting the camera to a light source.

In practical scenes with complex lighting, materials and geometry, the space of possible light paths is large and high-dimensional, making Monte Carlo based rendering methods the predominant solution technique [Pharr et al. 2016]. Of these methods, path tracing [Kajiya 1986] in particular has found widespread adoption in industry [Christensen and Jarosz 2016; Fascione et al. 2017], owing to its simplicity and efficiency.

Although path tracing can perform well most of the time, it may undersample small sets of high-contribution light paths (such as caustics), which introduces excessive variance and leads to long render times before a noise-free image is obtained (Figure 2, middle row). More sophisticated Monte Carlo methods [Georgiev et al. 2012; Hachisuka et al. 2012; Lafortune and Willems 1993; Veach and Guibas 1994] are superior at sampling complex transport, but these methods come with considerable computational overhead. If the scene is dominated by "simple" transport, i.e. that which path tracing can handle well, these methods perform significantly worse than path tracing when compared at equal time. This is because these methods can only be employed *globally*, i.e. on all transport, rather than *locally*, i.e. only on complex transport where they provide an advantage.

Rendering algorithms based on Markov chains such as Metropolis light transport (MLT) [Veach and Guibas 1997] excel at focusing computational effort on complex transport by reusing and iteratively perturbing existing samples. While they can sometimes reduce variance significantly compared to pure Monte Carlo methods, they also explore the available space unevenly, leading to poor stratification, "splotchy" non-uniform noise and flickering artifacts in animations. This is particularly apparent on simple transport such as direct lighting, which would otherwise be sampled well by path tracing (Figure 2, bottom row). Implementations of these methods typically include hard-coded rules for separating out simple transport to be handled by a separate Monte Carlo method, but these rules do not generalize well. Much like sophisticated pure Monte Carlo methods, methods based on MLT only outperform path tracing when the scene is dominated by complex transport.

In this paper, we introduce a way to selectively combine traditional Monte Carlo rendering algorithms with Metropolis sampling to leverage the benefits of both. We partition the integration domain into regions that are handled well by a "base" Monte Carlo method (like path tracing) and those that are not. Whenever the base method encounters a sample in the latter region that would introduce high variance, the sample is used instead as a seed for a Markov chain that locally explores the difficult transport with a more sophisticated rendering algorithm (like bidirectional path tracing).

The design space for this type of algorithm is large, and we identify a particularly practical configuration that has the following important properties:

- **Robustness.** Our method is usually much better and never much worse than the base method. By quickly classifying which samples should be handled by the Markov chain and carefully allocating computational effort, our method does not degrade the performance of the base method and avoids adversely affecting render times when not needed.
- **Stratification.** We drastically reduce temporal flickering and improve stratification by restricting the Markov chain only to paths the base method cannot handle well. Low-discrepancy sequences and adaptive sampling may still be used for the base method, further improving noise characteristics.
- **Local use of complex methods.** Even if the base method is simple (e.g. path tracing), the Markov chain may use inversions [Bitterli et al. 2018; Otsu et al. 2017; Pantaleoni 2017] to employ more complex rendering methods (e.g. bidirectional path tracing) to explore nearby difficult transport. The ability to use these sophisticated algorithms locally only for samples that need it allows our algorithm to improve both upon the base method and its more sophisticated variants.

We motivate our combination of Monte Carlo and Markov chains with Multiple Importance Sampling [Veach and Guibas 1995] (section 3), and show that the standard MIS heuristics do poorly for our purposes (section 4). MIS for these estimators can be interpreted as a crude firefly detector, and we improve on the standard heuristics with a more sophisticated outlier detector. We then discuss the design decisions needed to make the resulting MLT estimator practical (section 5) and describe our choice of outlier detector (section 6).

2 RELATED WORK

Metropolis. Starting with the original Metropolis light transport algorithm [Veach and Guibas 1997], much effort has been invested in finding specialized perturbation strategies for complex transport configurations [Hanika et al. 2015; Jakob and Marschner 2012], visibility [Hachisuka and Jensen 2011; Šik et al. 2016], and improved step size control [Otsu et al. 2018; Zsolnai and Szirmay-Kalos 2013]. MLT algorithms based in primary sample space [Hachisuka et al. 2014; Kelemen et al. 2002] are of particular interest due to their simplicity and ability to leverage importance sampling strategies present in modern renderers. Recent extensions [Bitterli et al. 2018; Otsu et al. 2017; Pantaleoni 2017] allow these methods to leverage inverses to rapidly switch between different importance sampling techniques, which allows them to explore difficult paths more easily. Although in theory our algorithm could be used with any of these methods, we focus on primary sample space methods and in particular Reversible Jump MLT (RJMLT) [Bitterli et al. 2018], as it presents a good trade-off between robustness and simplicity. Energy Redistribution Path Tracing [Cline et al. 2005] (ERPT) aims to improve stratification by running many short chains in parallel, as opposed to the few, long-lived chains of traditional methods. Similar to ERPT, we use well-seeded, short-lived chains to improve stratification, but combine it with a partitioning of the integration domain so that the Markov chains only need to explore small, high-energy parts of path space, which further improves efficiency.

Path Guiding. Adaptive importance sampling has a long history in light transport, starting with photon maps [Jensen 1995] or spatial data structures [Lafortune and Willems 1995] used to guide path tracing. Generally speaking, these methods build local representations of the incoming radiance based on past samples, and use these representations to guide future samples. Classical machine learning methods [Dahm and Keller 2017; Hey and Purgathofer 2002; Müller et al. 2017; Reibold et al. 2018; Vorba et al. 2014], and more recently neural networks [Müller et al. 2018] have found widespread use for representing radiance and guiding samples. Although these methods are capable of concentrating samples in high-variance areas, they can only do so *after the fact*; until the distribution of radiance is learned, problematic samples may introduce excessive variance into the image that takes many subsequent samples to remove. In addition, these methods require complex spatial acceleration structures that do not adapt well to large scene scales or highly directional transport. In contrast, our method does not require an initial learning period or spatial data structures, and can remove high variance samples in an unbiased manner before they enter the estimate.

Filtering and Denoising. A different approach to handling variance in MC rendering is to blur problematic samples or directly remove them from the image. This can be in the form of selective blurring based on image variance [Rousselle et al. 2012; Xu and Pattanaik 2005] or geometric features [Rousselle et al. 2013], or more sophisticated methods based on regression [Bitterli et al. 2016; Moon et al.

2014] or neural networks [Bako et al. 2017; Chaitanya et al. 2017]. Zwicker et al. [2015] provides a comprehensive survey up to 2015. These methods can be powerful at reducing residual variance, but come at the cost of introducing significant bias. Gradient-domain formulations [Gruson et al. 2018; Kettunen et al. 2015; Lehtinen et al. 2013; Manzi et al. 2015] can also reduce and distribute error more evenly across the image either via a screened Poisson solver [Pérez et al. 2003] or as a form of control variates [Rousselle et al. 2016]. In their unbiased form, however, these tend to be outperformed by biased denoising approaches. Instead of removing variance as a post-process, a common alternative approach is to prevent excessive variance from entering the image in the first place, either by clamping the contribution of a sample to some maximum value [Fascione et al. 2017; Keller 1997], or by statistically detecting outliers and discarding [DeCoro et al. 2010] or reweighting [Zirr et al. 2018] them. These methods still introduce bias, but limit the use of filtering only to samples with excessive variance. Instead of outright removing problematic samples, we use them as starting points for a Markov chain which spreads the brightness of the sample to its neighboring pixels, resembling an unbiased form of blurring high variance samples. Our method could still benefit from standard denoising methods to remove residual variance, and is more amenable to filtering than standard MLT methods due to improved noise uniformity.

3 BACKGROUND

Monte Carlo Rendering. The path integral framework forms the basis of most current rendering algorithms and expresses the value of a pixel measurement I in terms of an integral over the space $\mathcal{P}$ of all light paths

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

where $f(\bar{\mathbf{x}})$ is the measurement contribution of light path $\bar{\mathbf{x}}$. Traditional Monte Carlo rendering algorithms approximate Eq. (1) using the estimator

$$I \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(\bar{\mathbf{x}}^{(i)})}{p(\bar{\mathbf{x}}^{(i)})}\,, \tag{2}$$

which averages the result of N random and independent light paths $\bar{\mathbf{x}}^{(i)}$ that were sampled with probability density $p(\bar{\mathbf{x}}^{(i)})$.

Markov Chain Monte Carlo (MCMC). Instead of generating paths independently, rendering algorithms based on MCMC generate a sequence of correlated paths, starting from an initial seed path $\bar{\mathbf{x}}^{(0)}$. At each step of the chain, a tentative path $\bar{\mathbf{y}}^{(i+1)}$ is sampled from a proposal distribution $T(\bar{\mathbf{x}}^{(i)} \to \bar{\mathbf{y}}^{(i+1)})$, which is typically a narrow distribution (e.g. a Gaussian) centered on the previous path. The tentative path is then probabilistically accepted as the new state $\bar{\mathbf{x}}^{(i+1)}$ of the chain with probability $r(\bar{\mathbf{x}}^{(i)} \to \bar{\mathbf{y}}^{(i+1)})$. If the tentative path is rejected, the chain remains at its current state $\bar{\mathbf{x}}^{(i)}$.

The acceptance probability is carefully crafted in tandem with a target distribution $C(\bar{\mathbf{x}})$ so that the distribution of states approaches the target distribution in the limit. To obtain an image, MLT splats a value of $f(\bar{\mathbf{x}}^{(i)})/C(\bar{\mathbf{x}}^{(i)})$ to the image at each step of the Markov chain. In the following, we will assume that the path contribution itself (or its luminance) are chosen as the target distribution i.e. $C(\bar{\mathbf{x}}) = f(\bar{\mathbf{x}}^{(i)})/c$ for some normalization constant c, although other choices are possible [Hoberock and Hart 2010]. If the path contribution is not scalar, we assume $f(\bar{\mathbf{x}})$ to be its luminance for brevity, and use $f_{\mathrm{RGB}}(\bar{\mathbf{x}})$ to explicitly refer to the spectral version.

Primary Sample Space MLT. Using MLT directly in path space [Veach and Guibas 1997] can be cumbersome, because it is difficult to evaluate and tailor the proposal distribution to the materials and geometry present in the scene. Primary Sample Space MLT (PSSMLT) [Kelemen et al. 2002] instead leverages an existing Monte Carlo rendering algorithm (like path tracing) and recasts it as a function $\bar{\mathbf{x}} = S(\mathbf{u})$ that maps points $\mathbf{u} \in \mathcal{U}$ in the random number hypercube $\mathcal{U} = [0,1]^k$—the *primary sample space*—to paths $\bar{\mathbf{x}}$. Eq. (1) can then be rewritten as an integral over the primary sample space

$$I = \int_{\mathcal{U}} f(\mathbf{u})\,\mathrm{d}\mathbf{u}\,, \tag{3}$$

where $f(\mathbf{u}) = f(S(\mathbf{u}))/p(S(\mathbf{u}))$ for brevity, and $p(S(\mathbf{u}))$ is both the path space probability density of sample $\mathbf{u}$ and the Jacobian determinant of function S. We can view traditional Monte Carlo methods as approximating this integral with

$$I \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(\mathbf{u}^{(i)})}{p(\mathbf{u}^{(i)})} = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{u}^{(i)})\,. \tag{4}$$

Because these methods generate samples uniformly at random, the sample density in primary sample space is simply $p(\mathbf{u}^{(i)}) = 1$.

PSSMLT uses this formulation by running the Markov chain in primary sample space instead of path space directly. Each state of the Markov chain is a vector of random numbers, and the underlying mapping S is used to evaluate the contribution of each state. PSSMLT vastly simplifies proposals, since they can be sampled from simple Gaussians in the unit hypercube, and it leverages the existing importance sampling performed by the underlying mapping.

Reversible Jumps. Applying PSSMLT to bidirectional path tracing is complicated by the fact that bidirectional path tracing consists of multiple distinct sampling techniques that each represent a different mapping S_j from primary sample space to paths. Recent work on probabilistic inverses [Bitterli et al. 2018; Otsu et al. 2017; Pantaleoni 2017] allows mapping a path $\bar{\mathbf{x}}_i$ generated with technique i back to a point in primary sample space via the inverse mapping $\mathbf{u} = S_i^{-1}(\bar{\mathbf{x}}_i)$. Coupled with the forward mapping, this allows jumping between points $\mathbf{u}, \mathbf{v}$ sampled with techniques i, j through the composite mappings $\mathbf{v} = S_j^{-1}(S_i(\mathbf{u}))$ or $\mathbf{u} = S_i^{-1}(S_j(\mathbf{v}))$. This allows PSSMLT to easily switch between sampling techniques during rendering, facilitating faster exploration.

Multiple Importance Sampling (MIS). When multiple Monte Carlo estimators exist for the same integral, each individual estimator may work better or worse than the other estimators in different parts of the integration domain. Instead of choosing a single estimator, a more robust solution is to use a weighted combination of all estimators with weights chosen such that the combined estimator has lower variance than any individual estimator. MIS [Veach and Guibas 1995] provides several different heuristics for deriving these weights based on the PDFs of the estimators.

Assuming there are two strategies with PDFs p_1 and p_2, the *balance heuristic* weights samples drawn from the first strategy using

$$w_1(\mathbf{u}) = \frac{n_1 \cdot p_1(\mathbf{u})}{n_1 \cdot p_1(\mathbf{u}) + n_2 \cdot p_2(\mathbf{u})}\,, \tag{5}$$

and analogously for the second strategy, where n_1 and n_2 are the number of samples drawn from the two strategies.

While the balance heuristic provides continuously varying weights, the *maximum heuristic* assigns a binary weight of 1 to the estimator with the highest PDF, e.g. for the first strategy

$$w_1(\mathbf{u}) = \begin{cases} 1 & \text{if } n_1 \cdot p_1(\mathbf{u}) > n_2 \cdot p_2(\mathbf{u})\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{6}$$

and analogously for the second strategy. Both the balance and the maximum heuristic can be proven to be optimal in the sense that the variance of the combined estimator is never much worse than the variance of the estimator with optimal MIS weights [Veach 1997].

Discussion. In theory, MIS provides a mechanism for combining estimators with different strengths, and we could attempt to solve the problem laid out in Section 1 by coming up with a weighting heuristic that is aware of the difference in computational effort between estimators. However, this approach will fail in practice, since MIS only provides a way for *weighting* different estimators; it does not provide a way for *guiding* estimators towards paths that score a large weight. Naive MIS would simply waste any samples generated by a sophisticated estimator when they could be handled by a simple estimator; but what we need is to not generate these samples with an expensive estimator in the first place.

MLT is capable of guiding samples based on an arbitrary target function, and this function can incorporate MIS weights [Hachisuka et al. 2014]. We will use this idea and incorporate a specially crafted weight into the target distribution of a Markov chain, such that samples generated by computationally expensive estimators are focused where they provide a benefit. Methods based on MLT, however, suffer from non-uniform noise and temporal artifacts that we do not want to inherit, so we need to first identify the scenarios in which MLT performs well.

4 SELECTIVE METROPOLISATION VIA MIS

In Fig. 2, we illustrate a simplified direct lighting scenario with varying occlusion. A Monte Carlo approach without light sampling (middle row) in this scene performs well if large areas of uniform brightness of the emitter are visible (left), but does poorly if light is visible only through a small window (middle). In contrast, an MLT based method performs well if the target distribution is non-zero only in a narrow region (middle), but does poorly on large areas with uniform importance (left). Neither method performs well on a scene with a mixture of both cases (right).

We propose to formulate the selective combination of MC and MLT as a multiple importance sampling problem. The purpose of this section is to come up with an MIS weighting strategy $w(\mathbf{u})$ that assigns small weights to large, flat areas easily integrable with Monte Carlo and large weights to small islands that are best explored with MLT. We will then use the MC estimator to integrate $f(\mathbf{u})(1 - w(\mathbf{u}))$ in a straightforward way, and use MLT to explore

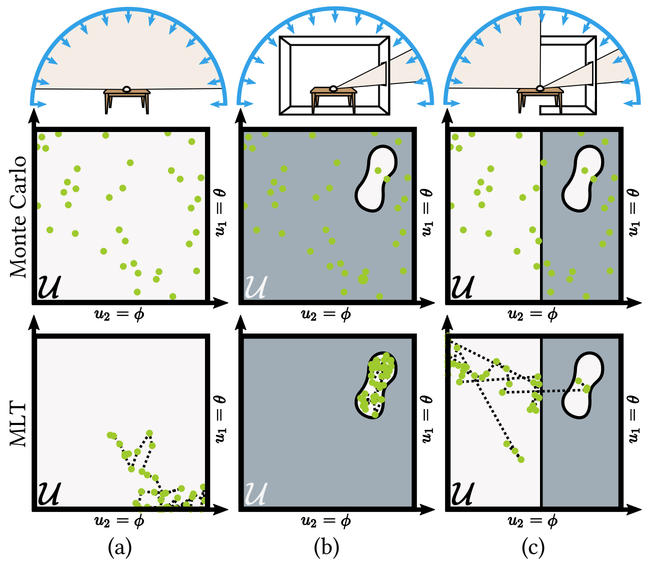

Fig. 2. We illustrate a simple scene lit by an environment emitter under different occlusion (top row). Monte Carlo methods (middle row) distribute their samples uniformly at random in primary sample space. This works well if the energy landscape is close to uniform (left column), but fails to sample small energy islands well (middle column). MLT (bottom row) performs poorly in flat regions (left column), but excels at exploring difficult paths (middle column). When a scene contains a mixture of both scenarios, neither strategy works well in isolation (right column).

the target distribution $f(\mathbf{u})w(\mathbf{u})$, which makes it focus its sampling effort where MLT performs well compared to MC.

We will first look at the standard weights derived from the balance- and maximum heuristic and explore why they are not ideal for our purposes, before deriving a better weighting scheme. For simplicity, we consider combining a Monte Carlo estimator with a PSSMLT method that uses the same underlying estimator as its mapping $S(\mathbf{u})$; however, the same weights hold for any MLT method in primary sample space.

Both the max- and balance heuristics require knowing the PDF of the two estimators. In primary sample space, the PDF of the Monte Carlo method is trivially $p_2(\mathbf{u}) = 1$. Following Kelemen et al. [2002], we use the target distribution as the PDF of the Markov chain, i.e. $p_1(\mathbf{u}) = f(\mathbf{u})/c$, and the MIS weight of the Markov chain becomes

$$w_1(\mathbf{u}) = \frac{n_1 \cdot (f(\mathbf{u})/c)}{n_1 \cdot (f(\mathbf{u})/c) + n_2 \cdot 1} = \frac{f(\mathbf{u})}{f(\mathbf{u}) + n_2/n_1 \cdot c} = \frac{f(\mathbf{u})}{f(\mathbf{u}) + b}\,, \tag{7}$$

for the balance heuristic, where b is some constant that does not depend on the sample, and

$$w_1(\mathbf{u}) = \begin{cases} 1 & \text{if } n_1 \cdot f(\mathbf{u})/c > n_2 \cdot 1 \\ 0 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } f(\mathbf{u}) > b' \\ 0 & \text{else}, \end{cases} \tag{8}$$

for the maximum heuristic and some other constant b'.

Equations (7) and (8) allow us to perform MIS between any Monte Carlo estimator and its PSSMLT variant. The Markov chain will automatically move towards regions where the base MC estimator performs poorly, and high-variance samples generated by the MC estimator will be downweighted before being added to the estimate.

This forms the basis of a workable rendering algorithm. However, these specific weights do not fully satisfy the goals set out at the beginning of this section, and we again turn to Fig. 2 for an intuition. For both heuristics, the MIS weight of the Markov chain depends solely on the brightness of the sample. If both the window and the large flat region have similar brightness, then these MIS weights will lead to either MLT or Monte Carlo being used for both regions, rather than the separation we would want. In practice, this leads to counter-intuitive results, such as MLT being used to explore bright direct lighting, and dim indirect lighting being left to Monte Carlo.

Although not ideal, the weight derived from the maximum heuristic provides a starting point for deriving a better weighting scheme: Looking at Eq. (8), this heuristic assigns zero weight to the Monte Carlo estimator whenever the sample weight exceeds some threshold, i.e. $f(\mathbf{u}) > b$. If we did not use MLT in addition, this scheme would be equivalent to discarding samples with high contribution ("fireflies") to avoid high variance at the cost of bias.

Classifying samples as fireflies based on brightness alone is the simplest in a family of *outlier detectors*. These algorithms detect unlikely Monte Carlo samples, and are usually employed as a last-resort measure to remove troublesome samples. However, outlier detectors form a strong candidate for a good weighting scheme for combining MLT and Monte Carlo: small islands in path space that are easily explored with MLT directly correspond to outlier samples, whereas samples in large uniform areas that are well sampled by MC would be classified as inliers.

Following these insights, we propose the following algorithm: Given an outlier detector that assigns large weights $w(\mathbf{u})$ to unlikely samples, we integrate the function $f(\mathbf{u}) \cdot (1 - w(\mathbf{u}))$ with a standard MC estimator, and use MLT to integrate the function $f(\mathbf{u}) \cdot w(\mathbf{u})$. We give a high-level overview of our method in algorithm 1. The Monte Carlo aspect of this algorithm is straightforward, but there is a large design space for the Markov chain. In the next section, we will explore the specific details needed to turn MLT into a practical and efficient rendering algorithm for our purposes.

Algorithm 1: High-level outline of our rendering algorithm

```
function renderOneSPP()
    // Run Monte Carlo sampler
    outliers ← {}
    for $i \leftarrow 0$ to numPixels do
        RNGState ← RNG
        samples ← traceMonteCarloPath(i, RNG)
        foreach $c \in$ samples do
            outlierDetectorSplat($i$, $c$)
            $w \leftarrow$ outlierWeight($i$, $c$)
            framebufferSplat($i$, $c \cdot (1 - w)$)
            if $w > 0$ then
                append(outliers, {$w \cdot c$, RNGState})
    // Bookkeeping
    $n_1 \leftarrow$ determineMLTSampleBudget(outliers)
    numChains ← $\frac{n_1}{\text{chainLength}}$
    // Run MLT
    for $i \leftarrow 1$ to numChains do
        seed ← selectSeed(candidates)
        runMLT(seed)
    // Apply changes to outlier detector from this iteration
    updateOutlierDetector()
```

5 PRACTICAL MLT ON SPARSE DISTRIBUTIONS

Naively running an existing MLT method on the target distribution $f(\mathbf{u}) \cdot w(\mathbf{u})$ works poorly in practice. By design, this target distribution is sparse and contains a large number of small, disconnected "islands". This creates two problems: Exploring the state space effectively poses a challenge, and finding initial states $\mathbf{u}^{(0)}$ with non-zero contribution is difficult.

Our solution to this problem is two-fold: First, we directly use discarded samples of the Monte Carlo estimator as initial states for MLT. MC samples with low MIS weights will have a large score in the corresponding MLT target distribution and therefore form suitable initial states for the Markov chain. This ensures a large and continually increasing sample pool for MLT to choose from. Secondly, because it is difficult for MLT to find its way across islands, we don't expend any effort into trying to do so. We only use local *perturbations* instead of global *mutations* that are likely to fail, and use many short-lived chains (100-1000 perturbations) instead of a single long-lived chain. The latter ensures we still get global coverage of the state space, and also greatly improves stratification and noise characteristics of MLT [Cline et al. 2005].

Each complete transport path formed by the Monte Carlo estimator (e.g. each successful shadow connection by a path tracer) is run through the outlier detector and recorded in an outlier pool if $w(\mathbf{u})$ is non-zero. For each chain, we randomly select one of the samples in the outlier pool as the initial state $\mathbf{u}^{(0)}$ and run the Markov chain for k steps. Assuming an MLT budget of n_1 samples, we may run $\lfloor n_1/k \rfloor$ chains in total. At each step $\mathbf{u}^{(i)}$ of the Markov chain, we record a value of

$$\frac{1}{n_1} \cdot \frac{f(\mathbf{u}^{(0)})}{\text{pmf}(\mathbf{u}^{(0)})} \cdot \frac{f_{\text{RGB}}(\mathbf{u}^{(i)})}{f(\mathbf{u}^{(i)})}, \tag{9}$$

in the framebuffer, where $f(\mathbf{u}^{(0)})$ is the brightness of the initial outlier, and $\text{pmf}(\mathbf{u}^{(0)})$ is the probability of picking that outlier as the initial state from the set of recorded outliers. To ensure all chains have approximately equal contribution, we set $\text{pmf}(\mathbf{u}^{(0)}) \propto f(\mathbf{u}^{(0)})$.

Similar to Cline et al. [2005], we did not find our setting of the chain length parameter to be critical. Shorter chains generally lead to increased correlation artifacts, while longer chains lead to increased noise as fewer seed paths can be explored. We use 1000 mutations as a sane default, and did not need to adjust it for different scenes.

5.1 Allocating Samples

To make our algorithm effective, we need to appropriately budget the number of samples that the MLT- and the Monte Carlo estimator should take. Too few samples for MLT means we do not improve much over the Monte Carlo estimator alone, and too many increases render times without much decrease in variance.

The weighting introduced by the outlier detector effectively creates a soft partitioning of the integrand into two domains, where

$w(\mathbf{u})$ is 1 or 0 respectively. The *ideal* sampling PDF is perfectly proportional to the integrand; therefore, it would on average place a number of samples in each domain proportional to the domain's total brightness. That is, if the total number of samples taken is n, then the region handled by MLT should receive

$$n_1 = n \cdot r \quad \text{with} \quad r = \frac{\int_{\mathcal{U}} w(\mathbf{u}) f(\mathbf{u})\, \mathrm{d}\mathbf{u}}{\int_{\mathcal{U}} f(\mathbf{u})\, \mathrm{d}\mathbf{u}} \tag{10}$$

samples to match the distribution of the ideal PDF. The Monte Carlo sampler will deviate from the ideal PDF, and will place a smaller fraction than r in the outlier domain. We use the MLT estimator to compensate for this imbalance by adding additional samples exclusively in the outlier region. Knowing that $n = n_1 + n_2$, the number of MLT samples is then determined by $n_1 = n_2 \cdot r/(1-r)$.

The number n_2 of MC samples is determined by the user (via samples-per-pixel), but we do not know the ideal sample ratio r *a priori*. However, we note that the unweighted MC samples are an estimator of the denominator of Equation 10, and the weighted samples an estimator of the numerator. We therefore keep running averages of these two quantities and use their ratio to estimate r incrementally during rendering.

6 DETECTING OUTLIERS

In theory, our algorithm is not limited to any particular outlier detector. However, the choice of detector has a large influence on the practical efficiency of the algorithm, and if our goal is to obtain a fast renderer, this imposes several design constraints on the choice of outlier detector.

The primary goal of our algorithm is to not degrade the performance of the base integrator when it works well. In particular if path tracing is used as the base, the cost of obtaining a sample is comparably low; since the outlier classifier is run on each sample, the computational efficiency of the classifier is paramount. In addition, high-quality renderings require a large number of samples, and therefore the memory footprint of the classifier should be independent of the sample count. These constraints make outlier detectors operating in image space an attractive choice.

Pixels may contain samples from many independent integrals (multiple path lengths, light sources, etc.). Sample statistics for a pixel thus usually contain multiple modes, and the outlier detector should be able to represent these; simple approaches such as per-pixel mean and variance are not sufficient.

This initially led us to consider *Density-Based Outlier Detection (DBOR)* [DeCoro et al. 2010]. While DBOR is an acceptable choice for the purposes of our algorithm, it does not have bounded memory footprint and its computational cost causes considerable efficiency loss on simple scenes (such as e.g. a Cornell box). More recently, Zirr et al. [2018] introduced an outlier detector based on *cascaded* framebuffers. While similar in spirit to DBOR, this detector is computationally inexpensive, has bounded memory footprint, and is very effective at outlier detection, all the while being a very simple algorithm to implement. From a practical standpoint, this makes it an attractive choice for our algorithm.

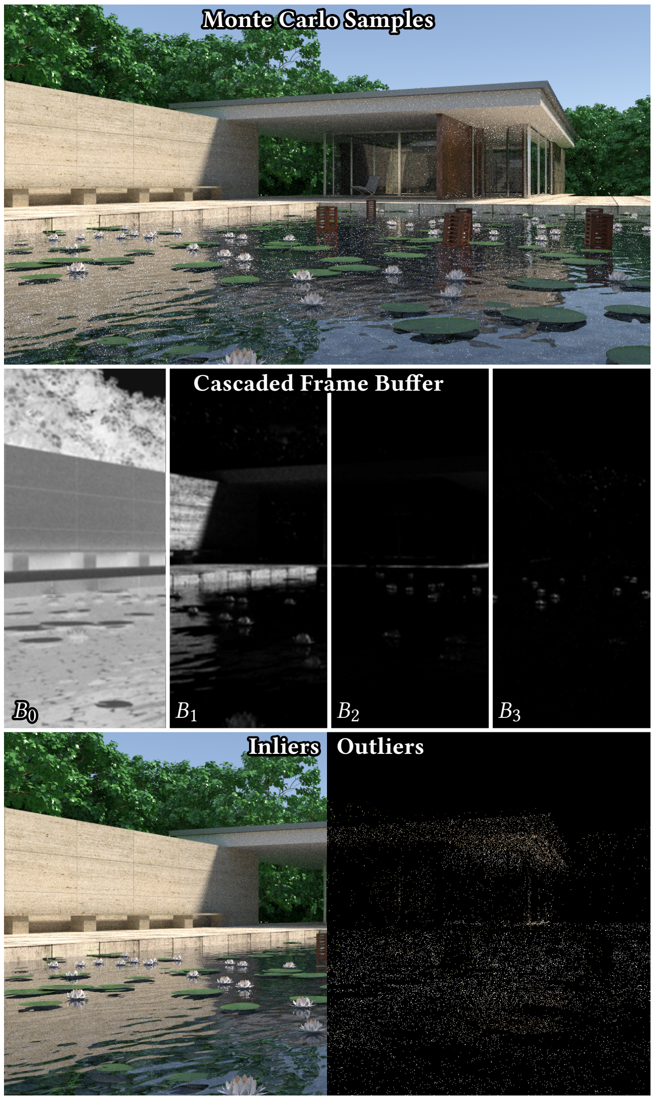

Fig. 3. We illustrate our chosen outlier detector in a scene containing difficult lighting. Starting from the raw Monte Carlo samples (top row), we first record individual samples in one of several cascaded frame buffers based on sample brightness (middle row). Using the number of samples with similar brightness as an indicator of whether a sample is unlikely, the detector reliably distinguishes paths handled well by Monte Carlo (bottom left) from outlier paths (bottom right) on a per-sample basis. Scene ©Hamza Cheggour

6.1 Implementation Details

In the following, we first describe the detector of Zirr et al. [2018] in detail, before motivating our particular choice of parameters. We illustrate this detector in Figure 3.

This detector stores a set of exponentially spaced framebuffers $B_0, \ldots, B_m$, in which buffer B_j at each pixel counts the number of samples that fall in a luminance range centered on $\alpha \cdot \beta^j$. All samples generated by the Monte Carlo estimator, outlier or not, are recorded in the cascaded framebuffer. For a sample with luminance x, we find the nearest buffers B_j and B_{j+1} such that $j = \lfloor \log_\beta x/\alpha \rfloor$, and add a value of $w = \log_\beta(x/\alpha) - j$ to the corresponding pixel in B_j and a value of $1 - w$ to B_{j+1}. To determine if a sample is an outlier, we linearly interpolate from the nearest buffers using the same weights as with splatting, and check if the interpolated value is smaller than a threshold τ.

Choice of Parameters. High-energy samples are almost exclusively outliers, and we consider samples far beyond meaningful image values (1000 in our implementation) automatically as outliers. The detector only needs to operate in the regime below this threshold, and we pick the number m of buffers to cover the luminance range below 1000.

The exponential spacing of the buffers controls the discretization of the luminance range: A smaller spacing means a larger number of modes can be represented per pixel; but it also means per-buffer statistics are noisier at lower sample counts. We choose ($\alpha = 1/2, \beta = 2$) as a tradeoff between robust detection at low sample counts and luminance resolution. This differs from the powers of 8 used by Zirr et al. [2018]; this is because they focus on extremely high-energy fireflies and require all samples to be splatted to a buffer, thus the larger dynamic range. Our method benefits from discriminating lower energy outliers and gracefully handles samples brighter than the maximum representable luminance, motivating the use of a smaller spacing.

We set the classification threshold $\tau = \max(3, \mathrm{spp}/m)$, which considers luminance ranges that receive less than uniform (spp/m) samples to likely contain outliers. At low sample counts we do not have much information; we err on the side of caution and clamp the threshold to a lower bound.

Updating the Detector. Updating the outlier detector while simultaneously using it to classify outliers would lead to bias, as MIS weights would shift during rendering. We instead render the image in iterations of a small number of samples per pixel at a time. We keep two copies of the outlier detector: The first is used only for classifying outliers and is left unchanged during each iteration, and the second records new samples generated from the Monte Carlo estimator. After the iteration is complete, we copy the contents of the second detector into the first.

Additional Thresholding. Regardless of which detector is used, it may classify samples as outliers that are too dim to be worth exploring with MLT. The likelihood of an outlier being picked as a chain seed depends on its brightness, and dim outliers are picked so rarely that it is better to splat them to the image than to hope for them to become a Markov chain. To account for this, we impose a minimum brightness threshold for a sample to be considered an outlier. The threshold is determined automatically during rendering: Between each iteration, we identify all outliers whose probability of being picked for MLT exploration is smaller than some threshold (1/2 in our implementation), and set the maximum sample brightness within this set as the threshold for the next iteration.

7 RESULTS

We implemented our method on top of an open source rendering system, and tested it on a diverse test set of indoor- and outdoor scenes that contain varying mixtures of difficult and simple light transport. Although our method could be used with any Monte Carlo and MLT estimator, we found the combination of path tracing and RJMLT to work the best, and only show results using this combination in the paper. We include additional results using bidirectional path tracing combined with RJMLT in the supplemental. We use the public implementation provided by the authors of RJMLT for comparison, and will release the full source code of our method. We rendered time-sensitive results on an 8-core AMD Ryzen 1800x with 16 threads, and all remaining results on a heterogeneous compute cluster. We evaluate our method using a variety of metrics, detailed in the following paragraphs.

Equal-time Renderings. In Fig. 1 and Fig. 4, we show several scenes rendered with path tracing, RJMLT, our method and a modified version of ERPT that uses RJMLT. Sample counts were normalized so all methods complete in equal time of 150 seconds. We also show the MSE of each method to a reference, relative to our method. This corresponds to how much longer we should expect to render with each method to obtain the same error as our method. In nearly all scenes, our method provides improvements of 2–15× over the estimators our method is combining, both visually and in terms of MSE. The only exception is the pool scene: Because the image is dominated by very difficult paths, the computational effort expended on path tracing does not provide much benefit over running RJMLT alone, and our method performs slightly (0.8×) worse than RJMLT.

Convergence Plots. For MLT based methods, individual renderings may not be representative of the behavior of the algorithm due to correlations. For each estimator, we therefore also rendered 30 independent runs with different random seeds, and computed the evolution of MSE over time. The average MSE then provides a more stable comparison metric, which we plot in Fig. 5. We also compute the standard deviation of the MSE across all runs, which gives an indication of the temporal flickering for each method. We visualize this with a shaded region around the average MSE corresponding to one standard deviation. Note that because of logarithmic axes, this region is asymmetric around the mean. As expected from a pure MLT method, RJMLT has erratic convergence and significant variance between runs, which makes it undesirable for practical use. ERPT has smoother convergence and less temporal variance than RJMLT, but converges significantly slower. Much like path tracing, our method provides smooth convergence and temporal stability, but with significantly lower average error. In most scenes, our method significantly outperforms RJMLT, except when the scene is dominated by complex transport; however, we offer significantly more stable convergence and still outperform path tracing.

Temporal Videos. To visually aid in comparing temporal flickering, we also provide videos for each method showing 30 independent runs in quick succession. We refer to the supplemental material containing an interactive viewer for comparing these videos.

8 CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

We introduced a new rendering algorithm that selectively combines different Monte Carlo estimators through MLT. Using MIS as a theoretical foundation for combining MC estimators with different strengths, we incorporate a special MIS weight into the target distribution of a Markov chain in order to focus its sampling effort in

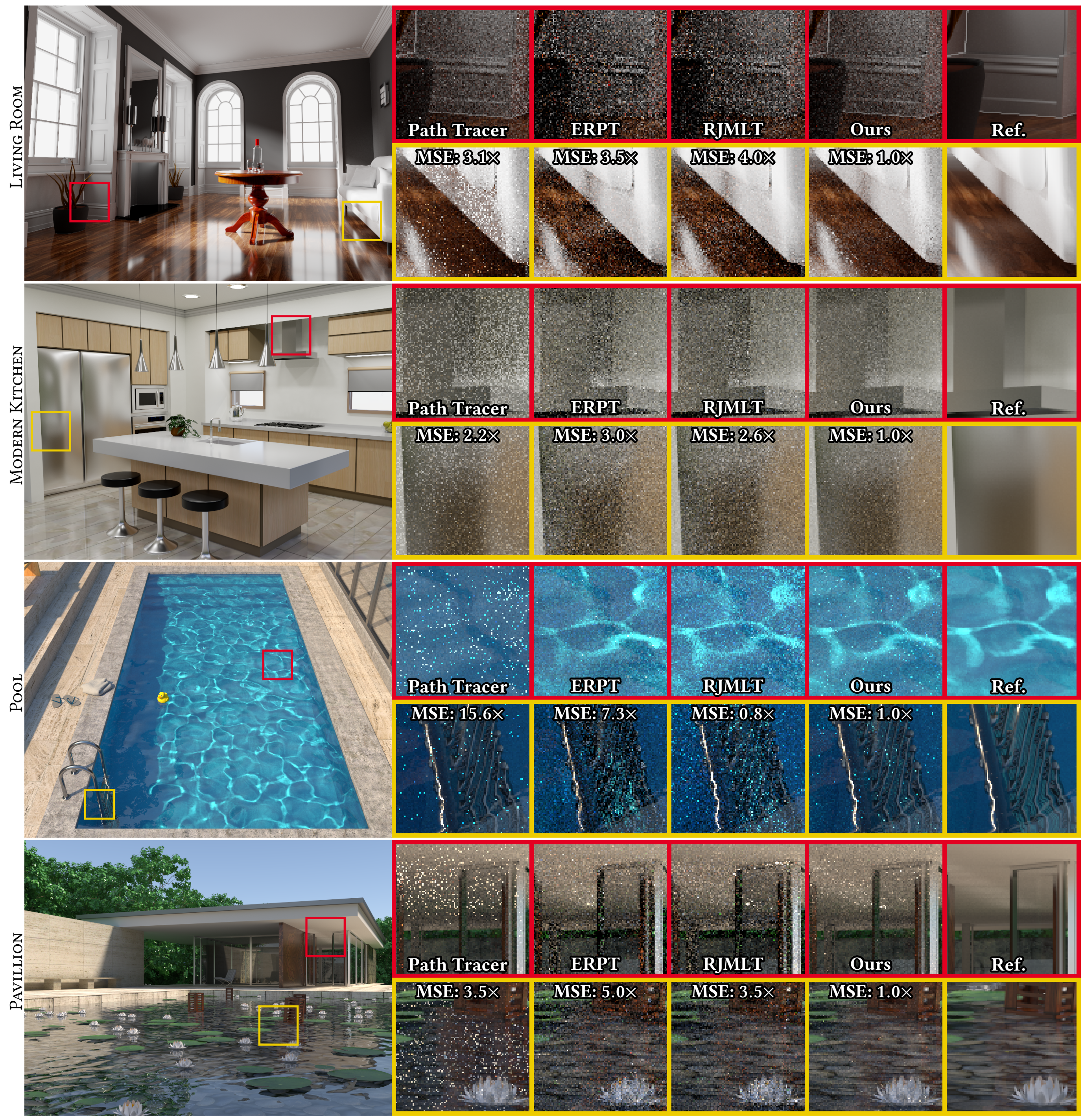

Fig. 4. We show a diverse set of scenes rendered with path tracing, ERPT with reversible jumps, RJMLT and our method, and compare them at equal render time. We show the relative MSE (computed over the entire image) of each method compared to ours, which is equivalent to how much longer each method would have to render to reach the same MSE as ours. Our method shows significant noise reduction compared to prior work. Please see the supplemental material for full-size images and temporal comparisons. Living room scene ©Wig42. Kitchen scene ©TheCGNinja. Pool scene ©Jiří Vorba. Pavillion scene ©Hamza Cheggour

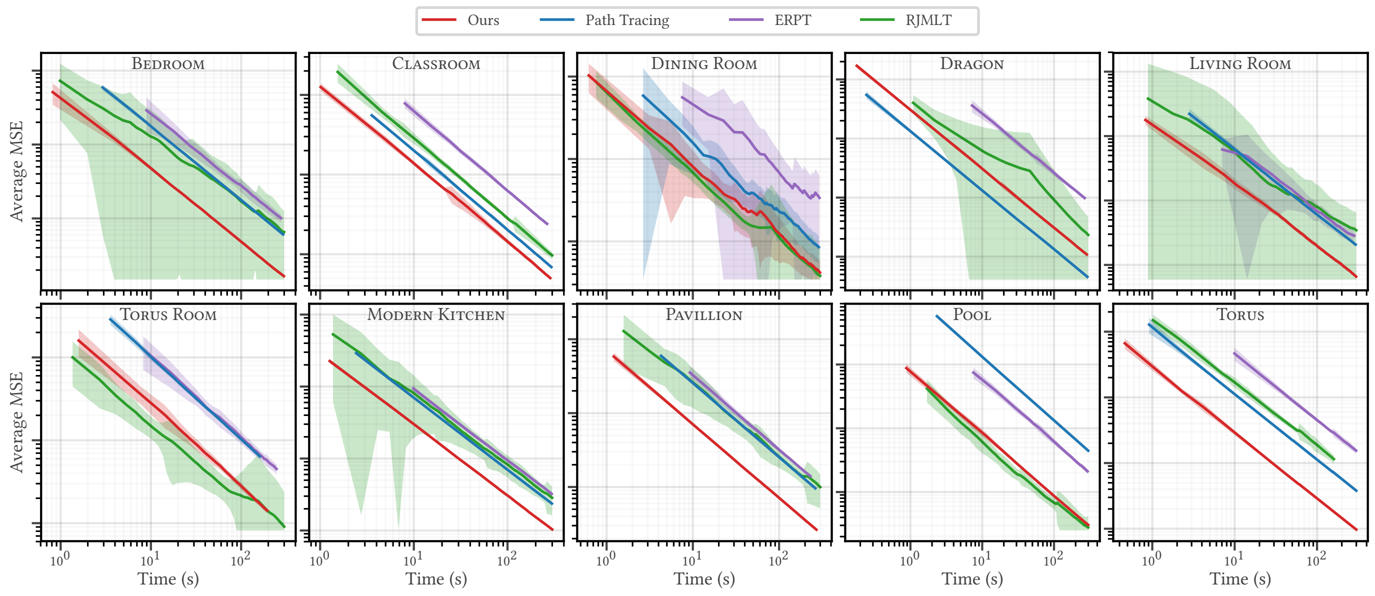

Fig. 5. We measure the MSE of our method and previous work over 30 independent runs and visualize both the mean MSE (thick curve) as well as the standard deviation of the MSE (shaded regions) over equal time. A large shaded region means that the MSE fluctuates significantly between runs, indicative of severe temporal flickering. On most scenes, our method has both significantly lower MSE as well as less variation across runs compared to previous work. Note that because of logarithmic axes, the shaded region is not symmetric around the mean. We focus on relative performance and omit y-axis labels for space reasons.

regions not handled well by a base MC estimator. We show that the balance- and maximum heuristic reduce to functions solely based on the sample brightness in primary sample space, which turns the maximum heuristic into a crude firefly detector. Based on these insights, we alleviate the problems of the standard weights by using a more reliable outlier detector. We discuss important problems of the Markov chain initialization and parameter design to obtain a robust and efficient rendering algorithm. Our final method significantly outperforms both path tracing and RJMLT on the majority of our test scenes, with significantly reduced temporal flickering and improved noise characteristics compared to standard MLT.

There are some limitations to our work. For example, we rely on a simple MC estimator to produce samples for the Markov chain to explore. If the transport is too difficult for the MC estimator to sample, then it will not produce enough seed samples for the MLT portion of our method to work effectively. This problem can be circumvented by using a better MC base method (like bidirectional path tracing), but this defeats our goal of avoiding experimentation with different integrators or parameters for each scene.

The Dragon scene represents a failure case of our method. Here, our method spends computational effort exploring glossy-glossy interreflections, which slightly reduces noise but disproportionately increases render times, and our method performs approximately 0.5× as well as path tracing. We did not do parameter tuning or much optimization on our method, and it is possible that may close this gap. Better perturbation strategies or better criteria for which samples to explore might also improve the performance of our method on the simplest of scenes. However, our method still performs reliably on scenes with more realistic complexity.

There are several ways in which our work could be extended. We only explored unbiased estimators in this paper, but one could also combine the Markov chain with biased methods such as photon mapping [Jensen 2001] or VCM/UPS [Georgiev et al. 2012; Hachisuka et al. 2012] to selectively introduce bias where the base MC estimator fails. Combinations of MLT with photon mapping [Hachisuka and Jensen 2011] or VCM/UPS [Šik et al. 2016] have been attempted previously and could find direct application in our work.

Although our outlier-based weighting scheme works well in practice, a more principled derivation would be interesting. The variance proofs of the balance- and maximum heuristic only hold under sample independence, and it would be instructive to derive new heuristics that explicitly incorporate not just the PDFs, but also the sample correlations and computational cost to minimize variance. These weights may be able to more readily explain when MLT works better than standard Monte Carlo, and when it does not.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation (Grants CNS-1205521 and IIS-1812796).

REFERENCES

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36, 4 (July 2017), 1–14. https://doi.org/10/gbxfbt

Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. 2018. Reversible Jump Metropolis Light Transport Using Inverse Mappings. *ACM Transactions on Graphics* 37, 1 (Jan. 2018), 1:1–1:12. https://doi.org/10/gd52ph

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-Order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 35, 4 (June 2016), 107–117. https://doi.org/10/f842kc

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder.

ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36, 4 (July 2017), 98:1–98:12. https://doi.org/10/gbxhcv

Per H. Christensen and Wojciech Jarosz. 2016. The Path to Path-Traced Movies. *Foundations and Trends® in Computer Graphics and Vision* 10, 2 (Oct. 2016), 103–175. https://doi.org/10/gfjwjc

David Cline, Justin Talbot, and Parris Egbert. 2005. Energy Redistribution Path Tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 24, 3 (July 2005), 1186–1195. https://doi.org/10/b3xtrn

Ken Dahm and Alexander Keller. 2017. Learning Light Transport the Reinforced Way. In *ACM SIGGRAPH Talks*. ACM Press, Article 73, 2 pages. https://doi.org/10/gfzsm4

Christopher DeCoro, Tim Weyrich, and Szymon Rusinkiewicz. 2010. Density-Based Outlier Rejection in Monte Carlo Rendering. *Computer Graphics Forum (Proceedings of Pacific Graphics)* 29, 7 (Sept. 2010), 2119–2125. https://doi.org/10/dsjs9s

Luca Fascione, Johannes Hanika, Marcos Fajardo, Per Christensen, Brent Burley, Brian Green, Rob Pieké, Christopher Kulla, Christophe Hery, Ryusuke Villemin, Daniel Heckenberg, and André Mazzone. 2017. Path Tracing in Production (Parts 1 and 2). In *ACM SIGGRAPH Course Notes*. https://doi.org/10/gfz2ck

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (Nov. 2012), 192:1–192:10. https://doi.org/10/gbb6q7

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-Domain Volumetric Photon Density Estimation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 37, 4 (July 2018), 82:1–82:13. https://doi.org/10/gd52p6

Toshiya Hachisuka and Henrik Wann Jensen. 2011. Robust Adaptive Photon Tracing Using Photon Path Visibility. *ACM Transactions on Graphics* 30, 5 (Oct. 2011), 114:1–114:11. https://doi.org/10/fpwzq9

Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. 2014. Multiplexed Metropolis Light Transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33, 4 (July 2014), 100:1–100:10. https://doi.org/10/f6cswv

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (Jan. 2012), 191:1–191:10. https://doi.org/10/gbb6n3

Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. 2015. Improved Half Vector Space Light Transport. *Computer Graphics Forum* 34, 4 (July 2015), 65–74. https://doi.org/10/gfzv83

Heinrich Hey and Werner Purgathofer. 2002. Importance Sampling with Hemispherical Particle Footprints. In *Proceedings of the Spring Conference on Computer Graphics (SCCG)*. ACM, New York, NY, USA, 107–114. https://doi.org/10/fmx2jp

Jared Hoberock and John C. Hart. 2010. Arbitrary Importance Functions for Metropolis Light Transport. *Computer Graphics Forum* 29, 6 (Sept. 2010), 1993–2003. https://doi.org/10/dttgfv

David S. Immel, Michael F. Cohen, and Donald P. Greenberg. 1986. A Radiosity Method for Non-Diffuse Environments. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (Aug. 1986), 133–142. https://doi.org/10/dmjm9t

Wenzel Jakob and Steve Marschner. 2012. Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 31, 4 (July 2012), 58:1–58:13. https://doi.org/10/gfzq4p

Henrik Wann Jensen. 1995. Importance Driven Path Tracing Using the Photon Map. In *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer-Verlag, 326–335. https://doi.org/10/gf2hcr

Henrik Wann Jensen. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Ltd., Natick, MA, USA.

James T. Kajiya. 1986. The Rendering Equation. *Computer Graphics (Proceedings of SIGGRAPH)* 20, 4 (Aug. 1986), 143–150. https://doi.org/10/cvf53j

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum* 21, 3 (Sept. 2002), 531–540. https://doi.org/10/bfrsqn

Alexander Keller. 1997. Instant Radiosity. In *Annual Conference Series (Proceedings of SIGGRAPH)*. ACM Press, 49–56. https://doi.org/10/fqch2z

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 34, 4 (July 2015), 123. https://doi.org/10/gfzrhn

Eric P. Lafortune and Yves D. Willems. 1993. Bi-Directional Path Tracing. In *Proceedings of the International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*, H. P. Santo (Ed.), Vol. 93. Alvor, Portugal, 145–153.

Eric P. Lafortune and Yves D. Willems. 1995. A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer-Verlag, NY, 11–20. https://doi.org/10/gfz5ns

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 32, 4 (July 2013), 95:1–95:12. https://doi.org/10/gbdghd

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing. In *Proceedings of EGSR (Experimental Ideas & Implementations)*. https://doi.org/10/gf2hcw

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Transactions on Graphics* 33, 5 (Sept. 2014), 170:1–170:14. https://doi.org/10/f6km7m

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 36, 4 (June 2017), 91–100. https://doi.org/10/gbnvrs

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *arXiv:1808.03856 [cs, stat]* (Aug. 2018). arXiv:cs, stat/1808.03856

Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. 2018. Geometry-Aware Metropolis Light Transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (2018), 278:1–278:11. https://doi.org/10/gf2r3t

Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. 2017. Fusing State Spaces for Markov Chain Monte Carlo Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36, 4 (July 2017), 74:1–74:10. https://doi.org/10/gbxjs9

Jacopo Pantaleoni. 2017. Charted Metropolis Light Transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 36, 4 (July 2017), 75:1–75:14. https://doi.org/10/gfzq78

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 22, 3 (July 2003), 313–318. https://doi.org/10/bdk2h9

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3 ed.). Morgan Kaufmann, San Francisco, CA, USA.

Florian Reibold, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. 2018. Selective Guided Sampling with Complete Light Transport Paths. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (Dec. 2018), 223:1–223:14. https://doi.org/10/gf2g93

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-Space Control Variates for Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (Nov. 2016), 169:1–169:12. https://doi.org/10/f9cphw

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-Local Means Filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 31, 6 (Nov. 2012), 195:1–195:11. https://doi.org/10/f96zx3

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising Using Feature and Color Information. *Computer Graphics Forum (Proceedings of Pacific Graphics)* 32, 7 (Oct. 2013), 121–130. https://doi.org/10/gfzwbn

Martin Šik, Hisanari Otsu, Toshiya Hachisuka, and Jaroslav Křivánek. 2016. Robust Light Transport Simulation via Metropolised Bidirectional Estimators. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (Nov. 2016), 245:1–245:12. https://doi.org/10/gfz4kj

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Thesis. Stanford University, United States – California.

Eric Veach and Leonidas J. Guibas. 1994. Bidirectional Estimators for Light Transport. In *Photorealistic Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)*, Georgios Sakas, Peter Shirley, and Stefan Müller (Eds.). Springer-Verlag, 145–167. https://doi.org/10/gfznbh

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Annual Conference Series (Proceedings of SIGGRAPH)*, Vol. 29. ACM Press, 419–428. https://doi.org/10/d7b6n4

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *Annual Conference Series (Proceedings of SIGGRAPH)*, Vol. 31. ACM Press, 65–76. https://doi.org/10/bkjqj4

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 (Proceedings of SIGGRAPH)* 33, 4 (Aug. 2014), 101:1–101:11. https://doi.org/10/f6c2cp

Ruifeng Xu and Sumanta N. Pattanaik. 2005. A Novel Monte Carlo Noise Reduction Operator. *IEEE Computer Graphics & Applications* 25, 2 (March 2005), 31–35. https://doi.org/10/cgs6vx

Tobias Zirr, Johannes Hanika, and Carsten Dachsbacher. 2018. Re-Weighting Firefly Samples for Improved Finite-Sample Monte Carlo Estimates. *Computer Graphics Forum* 37, 6 (Sept. 2018), 410–421. https://doi.org/10/gdv89k

Károly Zsolnai and László Szirmay-Kalos. 2013. Automatic Parameter Control for Metropolis Light Transport. In *Proceedings of Eurographics Short Papers*, M.-A. Otaduy and O. Sorkine (Eds.). The Eurographics Association. https://doi.org/10/gf2r3s

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (Proceedings of Eurographics State of the Art Reports)* 34, 2 (May 2015), 667–681. https://doi.org/10/f7k6kj