Efficient Image-Space Shape Splatting for Monte Carlo Rendering

XIAOCHUN TONG, University of Waterloo, Canada
TOSHIYA HACHISUKA, University of Waterloo, Canada

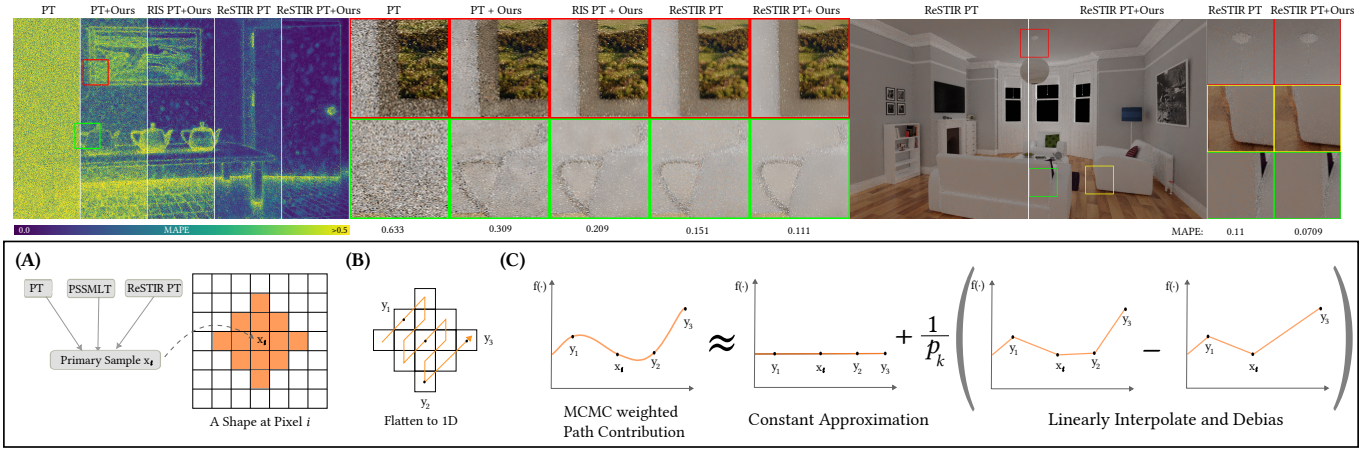

Fig. 1. We propose a framework for efficiently reusing light paths to pixels arranged in arbitrary 2D shapes in image space, each defined by multiple pixels. Our approach enables such reuse with a *sublinear* cost relative to the number of points in the shape. Top row: Our method can be implemented on top of point splatting methods such as PT, PSSMLT, or ReSTIR PT. (Left) We present equal time offline comparison, where our method consistently outperforms its single-pixel counterpart. (Right) When combined with ReSTIR PT, our method also suppresses color noise. Bottom row: Our method involves: (A) generating a primary light path using a baseline sampling method; associating each pixel with a 2D shape and (B) flattening it to 1D by running space-filling curve throughout the shape; (C) unbiasedly estimating weighted path contribution over the shape using a telescoping sum debiasing estimator [Misso et al. 2022]. We approximate the contribution with a constant function, then debias by evaluating one term of the telescoping sum with probability p_k, sampling k pixels within the shape, linearly interpolating them, and computing the difference by excluding one non-center pixel.

A typical Monte Carlo rendering method contributes one light path only to a single pixel at a time. Reusing light paths across multiple pixels, however, can amortize the cost and improve the efficiency. The state of the art of path reuse is to employ shift mapping to reduce the cost of path reuse, while its computation cost is still proportional to the number of pixels processed in shift mapping. We propose a general framework for efficiently reusing light paths to multiple pixels arranged in arbitrary two-dimensional shapes. Our shape is defined as a set of multiple pixels, and the framework allows us to reuse light paths among pixels in a shape faster than simply evaluating all pixels via shift mapping. The key idea is to sparsely evaluate the contribution of shifted paths at random pixels within the shape and interpolate the contribution to the other pixels. We apply a debiasing estimator to ensure unbiasedness. Our method can be integrated with many existing rendering methods and brings consistent improvement over its single-pixel counterpart.

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

Authors' addresses: Xiaochun Tong, University of Waterloo, Waterloo, Canada, xtong@uwaterloo.ca; Toshiya Hachisuka, University of Waterloo, Waterloo, Canada, toshiya.hachisuka@uwaterloo.ca.

ACM 0730-0301/2024/12-ART233
https://doi.org/10.1145/3687943

Additional Key Words and Phrases: physically-based rendering, light transport, MCMC, path reuse, debiasing

ACM Reference Format:
Xiaochun Tong and Toshiya Hachisuka. 2024. Efficient Image-Space Shape Splatting for Monte Carlo Rendering. *ACM Trans. Graph.* 43, 6, Article 233 (December 2024), 11 pages. https://doi.org/10.1145/3687943

1 INTRODUCTION

Monte Carlo rendering samples light paths between light sources and sensors to estimate the expected values of the sensor responses. Rendering thus requires sampling contributing light paths for every pixel (sensor) in the image. For example, path tracing [Kajiya 1986] samples a path that goes through each pixel from the sensor side, and repeats this process for all the pixels multiple times. The path integral formulation [Veach 1997] models this process as considering a single point on the image plane for each sampled light path and accumulating its contribution to the relevant pixels. Almost all existing Monte Carlo rendering methods build upon this concept, and each path sample can be thought as a *point* sample on the image.

We propose a framework which extends this correspondence between a light path and a single point to *multiple* points on the image. Since such multiple points often corresponds to neighboring pixels that form a certain shape (e.g., a square centered at each pixel), we refer to them as *shapes* in this paper. While it may be tempting to think that path reuse [Bauszat et al. 2017; Bekaert et al. 2002; Lin et al. 2022] already addresses this task, path reuse only achieves to

generate multiple paths at an amortized cost per sample, and its computational cost is still proportional to the number of points in the shape.

Our framework instead fundamentally reduces the computational cost of sampling a shape to *sublinear* to the number of points in the shape, making splatting shape samples an effective replacement of point splatting. The key idea is to approximate the contributions of all the points in a shape using a biased yet cheap estimator and apply debiasing [Misso et al. 2022] to remove the bias. By carefully constructing such a debiasing estimator, we can compute the unbiased contributions of a shape of ≈ 60 pixels by generating only 4 point samples on average, for instance. We formulate shape sampling and splatting as a multi-proposal MCMC step and weight the contributions according to MCMC acceptance probabilities. Our MCMC formulation allows us to effectively combine path samples from all the overlapping shapes without relying on multiple importance sampling which would otherwise incur a quadratic computation cost. We demonstrate the effectiveness of our framework by adding shape splatting to various Monte Carlo rendering methods. Shape-splatting variants of those methods consistently outperform the point-splatting counterparts, and can be added even on top of path reuse. Our technical contributions are a debiasing estimator for unbiased shape splatting with sublinear cost to the shape size, and an MCMC formulation of shape splatting that allows a fast evaluation of the sample weights with lower variance than pairwise MIS.

2 RELATED WORK

Line segment sampling. Line segments are one of the simplest shapes other than points. Previous work has explored accelerating Monte Carlo rendering by either generating segments in the image space or analytically integrating over line segments. Segment samples were used to compute analytic aliasing [Jones and Perry 2000], motion blur [Gribel et al. 2011, 2010], depth-of-field [Tzeng et al. 2012]. Barringer et al. [2012] employed line samples to compute accurate visibility of thin curves. Sun et al. [2013] constructed line segment samples with blue-noise properties using frequency analysis. Shirley and Wyman [2017] generated stratified 2D segments by wrapping stratified points in 2D unit squares. Singh et al. [2017] analyzed the variance and convergence properties of line and segment samples. Our work focuses on sampling a set of 2D points arranged in a specified manner (shape) based on a point sample, which has not been addressed before.

Path reuse. Bekaert style-path reusing [Bekaert et al. 2002] and its generalization [Bauszat et al. 2017] armotizes sampling cost by connecting eye subpaths to light subpaths of all other pixels in a tile. ReSTIR and Generalized RIS [Bitterli et al. 2020; Lin et al. 2022] amortize sampling cost by employing resampled importance sampling (RIS) to select light subpaths and reuse them spatio-temporally via shift mapping. As demonstrated by Lin et al. [2022], ReSTIR gained most of its advantage by using RIS to quickly approximate the target distribution, instead of simply reusing light paths via shift mapping.

The cost of path reuse is directly proportional to the number of shift mappings performed. If shift mapping is as costly or slower than generating a new sample from scratch, path reuse provides no benefit. We also use shift mapping to all the pixels within a shape, but we instead achieve a *sublinear* cost to the number of pixels within a shape. This property allows us to achieve some improvement even when shift mapping is slow or as costly as sampling a new path. We can also apply our shape splatting to the existing path reuse methods to further improve its efficiency.

Gradient-domain rendering. Gradient-domain rendering (GDR) [Gruson et al. 2018; Kettunen et al. 2015; Lehtinen et al. 2013; Manzi et al. 2015; Sun et al. 2017] computes image gradient by a pair of correlate paths sampled at adjacent pixels. GDR can also be interpreted as a form of path reuse since a sampled path will contributed to multiple pixels through shift mapping and Poisson reconstruction. Our method does not rely on a separate reconstruction step, and it can be seen as yet another form of path reuse besides GDR.

3 MOTIVATION

Let us go over a simplified example of sampling shapes from a given 2D distribution to establish the concept of shape splatting in this paper. Figure 2 shows this example where we are given a PDF $p(x, y)$ in the 2D space of (x, y) as an image and (x, y) represents a 2D pixel index. The variables (x, y) are discrete since they are integers indexing pixels. Let us consider the problem of copying the image $I(x, y)$ by sampling from $p(x, y)$, which is equivalent to estimating a histogram of samples generated according to $p(x, y)$ as was also used by Cline and Egbert [2005] to explain MCMC.

Point samples. Samples are usually 2D *points* $(x_i, y_i) \propto p(x, y)$ and the histogram is estimated by counting the number of samples within each pixel. Due to the stochastic nature, any pixel can potentially have zero samples given a finite number of total samples. Figure 2 shows the results of this baseline, and one can see that there are many zero histogram bins (black pixels) when the number of samples is small, and each sample contributes to *one* pixel.

Shape samples. Instead of point samples, we propose to sample shapes that are still distributed according to the same PDF $p(x, y)$. We define a shape as a set of (usually consecutive) points. Figure 2 shows our results, where the shape is a rotated quad that contains 61 points (pixels). One can recognize individual shapes being "splatted" as samples when the number of samples is small. We use "points" and "pixels" interchangeably in this paper whenever feasible. If the cost of generating each shape sample is less than that of generating all the point samples within the shape, shape samples effectively increase the number of samples.

Our main technical contribution is a general framework to achieve efficient shape splatting. Our framework allows us to splat this shape sample of 61 points in Figure 2 at the cost only of 4 point samples on average. Shape splatting is also more efficient than point splatting with an equal number of point samples taken (i.e., 4 times more for point samples) as Figure 2 demonstrates.

4 SHAPE SPLATTING

We first explain our method for splatting 1D shapes on a 1D image, and then we will discuss generalization to 2D shapes in Sec. 4.4. Listing 1 is a pseudocode of our method. Consider an image I of N pixels. We focus on scalar pixel values in the following, and colors are handled similarly to what is commonly done in MC rendering.

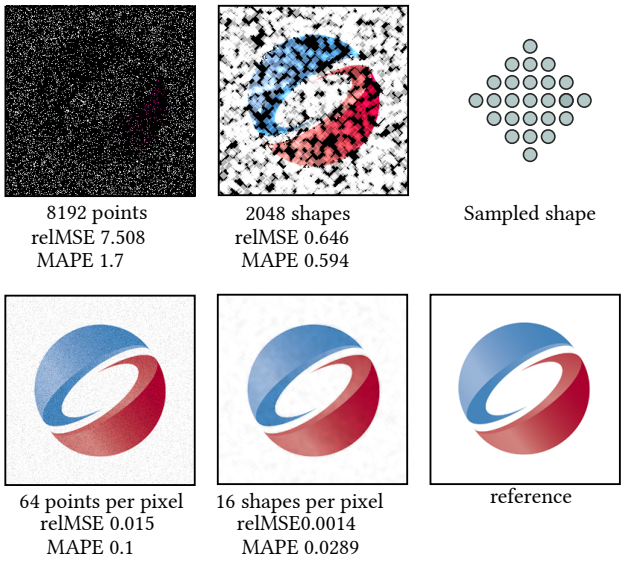

Fig. 2. We compare sampling a 2D distribution using points and shapes under the *equal point-sample counts*. The shape is a rotated quad with 61 points (pixels). Our method sample each shape with the cost of 4 point samples on average. Shift mapping in this example has the same cost as generating a new sample, so any gain is not coming from sample reuse. Shape samples fill in more pixels at the same cost, making it far more efficient than point sampling.

```python
# Unbiased shape splatting on a 1D image
# Input: pixel index i
# Output: An unbiased estimate of contribution
#         of the shape centered at i
def UnbiasedShapeSplatting(i):
    # 1. Sample the primary sample x from p(·)
    x, S = SamplePathAtPixel(i), ShapeAt(i)
    k, pmf_k = RandInt(1, |S|-1)

    # 2. Sample secondary pixels and path samples
    #    via shift mapping
    js = sort(
        {i} + sample k distinct pixels from S-{i})
    ys = [T_{i→j}(x) for j in js]
    # Compute MCMC weights that keeps p(·) invariant
    #  Eq. (7)
    ws = [MCMCWeights(x, y, |∂_x T_{i→j}(x)|) for y in ys]

    # 3. Debiasing. Sec. 4.3
    def C(x): return f(x) / p(x)
    D(1) = Ones(|S|) * C(x)
    D(k+1) = Interpolate(C(ys) * ws)
    res = D(1)
    # Eq. (10)
    for j′ in js:
        if j′ == i:
            continue
        js′ = js - {j′}
        ys′ = [T_{i→j}(x) for j in js′]
        ws′ = [MCMCWeights(x, y, |∂_x T_{i→j}(x)|) for y in ys′]
        D(k) = Interpolate(C(ys′) * ws′)
        res += (D(k+1) − D(k)) / (pmf_k * k) # Eq. (11)
    return res
```

Listing 1. Overview of our method. Note we demonstrate a naive implementation of debiasing here for simplicity. In practice, we cache the evaluation of $f(\cdot)$ and compute all differences between interpolations in a single pass.

Let us consider a more general problem than the example in Figure 2 where the histogram is proportional to an integral $I_i = \int g(i,\mathbf{x})d\mathbf{x}$ of a higher dimensional function $g(i,\mathbf{x})$. The example in Figure 2 corresponds to a case where the integrals I are directly given as an image. This generalized problem covers rendering with the path integral formulation where the pixel value I_i is determined by integrating a product of the pixel filter $h(i,\mathbf{x})$ and the contributions $f(\mathbf{x})$ of light transport paths $\mathbf{x}$ as

$$I_i = \int_{\mathcal{P}} g(i,\mathbf{x})d\mathbf{x} = \int_{\mathcal{P}_i} \underbrace{h(i,\mathbf{x})f(\mathbf{x})}_{g(i,\mathbf{x})}\, d\mathbf{x}. \tag{1}$$

The integration domain changes to $\mathcal{P}_i$ as a subset of the path space $\int_{\mathcal{P}}$ where $h(i,\mathbf{x}) > 0$. Let us consider $h(i,\mathbf{x}) = 1$ within the pixel i in the following so that we can simplify as $g(i,\mathbf{x}) = f(\mathbf{x})$ over $\mathcal{P}_i$. The vector of pixel values I is thus defined as

$$I = \begin{bmatrix} \vdots \\ I_i \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \int_{\mathcal{P}} g(i,\mathbf{x})dx \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \int_{\mathcal{P}_i} f(\mathbf{x})dx \\ \vdots \end{bmatrix}. \tag{2}$$

To estimate I, we first sample a pixel index i according to a discrete distribution $p_i(\cdot)$ and then sample a path $\mathbf{x} \sim p(\mathbf{x}|i)$. We refer to such $\mathbf{x}$ as *point samples*. While this formulation may sound incompatible with a typical scenario where we just loop over all the pixels and sample paths within each pixel (e.g., path tracing), it is equivalent to sample i from uniform distribution but with perfectly stratified sampling. The distribution of pixel index i and path $\mathbf{x}$ form a joint distribution $(i,\mathbf{x}) \sim p(\cdot)$. Since each path corresponds to a single pixel index i, we simply refer to $\mathbf{x} \sim p(\cdot)$ as sampling both the index and path. Once a path $\mathbf{x}$ is sampled at i, we would like to splat its contribution to other pixels such that the sum of all such shape splatting estimates I in an unbiased manner.

Shapes. We define a shape $\mathcal{S}$ as an *ordered* set of pixel indices $\mathcal{S} = \{i_1, ..., i_{|\mathcal{S}|}\} (i_1 < ... < i_{|\mathcal{S}|})$ where $|\mathcal{S}|$ is the number of points within the shape (e.g., 61 in Figure 2). We index each shape as $\mathcal{S}^i$ to enable a different shape per pixel. We denote $C(\mathcal{S}^i)_j$ as the contribution to the pixel j by the shape $\mathcal{S}^i$. We also refer to the pixel i as the *center pixel* of $\mathcal{S}^i$. The contribution to any other pixel that is not in the shape is zero. The exact definition of $C(\mathcal{S}^i)_j$depends on how shape splatting is performed. The only requirement is that the sum of $C(\mathcal{S}^i)_j$ of all pixel i unbiasedly estimates the image I

$$I_j = \sum_{i=1}^{N} C(\mathcal{S}^i)_j. \tag{3}$$

In each shape $\mathcal{S}^i$, we call the corresponding point sample that goes though the pixel i as the *primary* sample, and all the other point samples as the *secondary* samples.

Shape as Multiple Point Samples. A naive approach to sample a shape is to sample all the (primary and secondary) points within the shape by performing point sampling repeatedly. Each pixel value I_j

is estimated by taking an average of all the point samples from all the shapes that fall within the pixel j. The cost of sampling one shape in this naive approach is *equivalent* to sampling all the points within the shape, so this approach would *not* bring any benefit over point sampling with an equivalent number of samples. It nevertheless is useful as a baseline for our algorithm.

4.1 Amortized Shape Splatting via Mutations

Instead of sampling each point within a shape $\mathcal{S}^i$ from scratch, one can perform path reuse of the primary sample (which goes through i) to generate all the secondary samples within the shape. We focus on the contributions coming from a given shape $\mathcal{S}^i$ to all the other pixels (points) $\{i_1, ..., i_{|\mathcal{S}|}\}$ within it, which is equivalent to consider *splatting* of the contributions of each shape as opposed to *gathering* [Gharbi et al. 2019]. They are mathematically equivalent, but we found that splatting partially reused samples naturally maps to *mutation* in MCMC rendering.

Mutation involves generating a secondary sample $\{\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_{|\mathcal{S}|}}\}$ given the primary sample $\mathbf{x}_i$ (e.g, from a path tracer) and a distribution of mutated samples is called a proposal distribution. We choose a simple proposal distribution $T_i(j)$ that mutates pixel indices uniformly within the support of $\mathcal{S}^i$ at equal probabilities:

$$T_i(j) = 1/|\mathcal{S}^i| \cdot \mathbb{1}_{\mathcal{S}^i}(j) \tag{4}$$

Because $T_i(j)$ is a simple binary function, we can mutate the primary samples $\mathbf{x}_i$ to the all the other secondary samples *deterministically* by selecting each j in $\mathcal{S}^i$ exactly once over $|\mathcal{S}^i|$ proposals. While there are many different approaches to mutate or reuse path samples, we use shift mapping techniques developed for gradient-domain rendering [Bauszat et al. 2017; Kettunen et al. 2015; Lehtinen et al. 2013]. Due to the amortized cost per sample in mutation (or shift mapping), the cost of generating all the points within a shape is less than generating an equal number of point samples.

In general, secondary samples in the pixel j mutated to the pixel i will be distributed differently than samples within the pixel i. We need to take this difference into account to define a valid estimator of I_i. When all the pixels have the same shape $\mathcal{S}$ and the Jacobian of shift mapping from i to j is $\left|\partial_{\mathbf{x}} T_{i\to j}\right|$, the probability density of generating a sample from the shape at the pixel i to the pixel j is $p(\mathbf{x}|i)/\left|\partial_{\mathbf{x}} T_{i\to j}\right|$ which in general would be different from the probability density $p(\mathbf{x}|j)$ at the pixel j itself.

While this problem setting fits well with multiple importance sampling [Veach and Guibas 1995], the number of techniques here is $O(|\mathcal{S}^i|)$ and we found that the evaluation cost of the balance heuristic is impractical because each shape will cost $O(|\mathcal{S}^i|^2)$ for having $|\mathcal{S}^i|$ points and the balance heuristic for each point takes $O(|\mathcal{S}^i|)$. We propose an alternative based on a common practice of splatting proposals [Veach and Guibas 1997]. While we also experimented with a more practical alternative of pairwise MIS [Bitterli 2021], we found that our MCMC approach generates more accurate results, especially when combined with our debiasing estimator.

4.2 MCMC-based Weighting for Shape Splatting

By seeing each secondary sample as a proposal in MCMC, we can utilize the acceptance probability of the Metropolis-Hastings algorithm [Hastings 1970] to weight the contribution of each sample properly. In the Metropolis-Hastings algorithm, given a current state x, the proposal y is accepted according the probability $a(x \to y)$ designed to maintain the detailed-balance condition:

$$a(x \to y) = min\left(\frac{\pi(y)\mathcal{T}(y \to x)}{\pi(x)\mathcal{T}(x \to y)}, 1\right), \tag{5}$$

where $\mathcal{T}(x \to y)$ denotes the proposal distribution from state x to y. Given that $\mathcal{T}$ is ergodic, the Markov chain will, in the limit, attain the unique stationary distribution $\pi(x)$. A common practice in rendering is to accumulate the contributions coming from both the proposal and the current state regardless of the outcome of the acceptance of the proposal [Veach and Guibas 1997]. The contribution from the proposal y is weighted by $a(x \to y)$ and the contribution from the current state is weighted by $1 - a(x \to y)$. This technique would not introduce bias to the result because y is essentially replacing the stochastic process of accumulating either x or y alone by its mean. We utilize this technique to weight the contributions of all the points in an arbitrary shape sample.

In our current implementation via (invertible) shift mappings, the ratio $\frac{\mathcal{T}(y\to x)}{\mathcal{T}(i\to j)}$ turns out to be exactly $\left|\partial_{\mathbf{x}} T_{i\to j}\right|$. The acceptance probability, thus the weight, can be computed as:

$$a(\mathbf{x}_i \to \mathbf{x}_j) = min\left(\frac{p(\mathbf{x}_j)|\mathcal{S}^i|}{p(\mathbf{x}_i)|\mathcal{S}^j|}\left|\partial_{\mathbf{x}} T_{i\to j}\right|, 1\right), \tag{6}$$

with an additional $\frac{|\mathcal{S}^i|}{|\mathcal{S}^j|}$ factor to account for secondary pixels being sampled uniformly from shapes with different sizes. The contribution of each pixel in the shape can be estimated by computing the expected outcome of the accept decision accordingly:

$$w_{i\to j} = \begin{cases} \frac{1}{|\mathcal{S}^i|} a(\mathbf{x}_i \to \mathbf{x}_j) & (i \neq j) \\ \frac{1}{|\mathcal{S}^i|}(1 + \sum_{k\neq i} 1 - w_{i\to k}) & (i = j). \end{cases} \tag{7}$$

By mutating $\mathbf{x}_i$ to all pixels in $\mathcal{S}^i$ and weighting according to Eq. (7), we obtain a *pointwise* estimator $P(\mathcal{S}^i)$ of $C(\mathcal{S}^i)$.

$$\langle C(\mathcal{S}^i)_j \rangle = P(\mathcal{S}^i)_j = w_{i\to j}\frac{f(\mathbf{x}_j)}{p(\mathbf{x}_j)} \quad \text{where } \mathbf{x}_j = T_{i\to j}(\mathbf{x}_i) \tag{8}$$

Differences from MCMC. Unlike MCMC, we already have samples that are distributed according $p(\cdot)$ at each pixel. By setting the target as $\pi(\cdot) = p(\cdot)$, MCMC mutation keeps $p(\cdot)$ *invariant* because samples are already at the stationary distribution. The outcome of the acceptance of the proposal in our method is thus irrelevant because one can always generate a new sample (state) x that is already distributed according to $p(\cdot)$. We also have multiple proposals y from the same x because x corresponds to the primary sample and y correspond to all the secondary samples that are generated by perfectly stratified sampling of the proposal in Equation (4).

4.3 Efficient Shape Estimator via Debiasing

Our pointwise estimator $P(\mathcal{S}^i)$ Eq. (8) can reduce the cost of shape splatting without introducing any additional error. However, the overall splatting cost is nevertheless still $O(|\mathcal{S}^i|)$ since we need to evaluate the contribution of all the samples within the shape. To make our method more efficient, we would like to estimate $C(\mathcal{S}^i)$ as a whole at a *sublinear* cost to $|\mathcal{S}^i|$. We propose to employ *telescoping sum debiasing estimators* [Misso et al. 2022] to achieve this goal. Let

us focus on a single shape $\mathcal{S}^i$ to drop the superscript i for the rest of the discussion. Since all pixels $j \notin \mathcal{S}$ have zero contribution to the shape, we refer to only the pixels in the shape $j \in \mathcal{S}$ in the following. We also refer to the contribution $C(\mathcal{S})$ of $\mathcal{S}$ as an $\mathbb{R}^{|\mathcal{S}|}$ vector. One can reduce the cost of evaluating $C(\mathcal{S})$ by not sampling secondary samples but by replacing their contributions with the primary sample $P(\mathcal{S})_i$. The outcome of this is a biased estimator, which we refer to as $D(1)$ since it requires only 1 point sample. For a debiasing estimator, we first need to define a sequence of estimators $D(m)$ such that the bias $\mathbf{B}[D(m)] \to 0$ as m approaches a limit [Misso et al. 2022]. The limit in our case is $|\mathcal{S}|$ which samples all the points within the shape and thus $D(|\mathcal{S}|) = P(\mathcal{S})$. We now need to define $D(m)$ as a series of increasingly finer approximations of the unbiased estimator $P(\mathcal{S})$ such that the bias decreases with increasing m. Let $1 \le k \le |\mathcal{S}| - 1$ be a random integer with probability p_k. We sample an ordered subset $\mathcal{K} \subset \mathcal{S}, i \in \mathcal{K}$ of $k+1$ pixels, including the center and the other k pixels. A pointwise estimator $P(\mathcal{K})$ for this subset $\mathcal{K}$ is defined as $P(\mathcal{K})_j = P(\mathcal{S})_j$ for $j \in \mathcal{K}$ and 0 otherwise. Note that this estimator costs only $O(|\mathcal{K}|)$ because there are only $|\mathcal{K}|$ points to be evaluated. To approximate $P(\mathcal{S})$ with $P(\mathcal{K})$, we employ an interpolation operator $A_{\mathcal{S}} : R^k \to R^{|\mathcal{S}|}$ that fills in missing pixels in $P(\mathcal{K})$, thus we a have $A_{\mathcal{S}}(P(\mathcal{K})) = P(\mathcal{S})$ when $\mathcal{K} = \mathcal{S}$. We express $P(\mathcal{S})$ as a telescoping sum [Misso et al. 2022]

$$P(\mathcal{S}) = D(1) + \sum_{k=1}^{|\mathcal{S}|-1} \Delta D(k) \tag{9}$$

where $\Delta D(k) = D(k+1) - D(k)$ and $D(k) = A_{\mathcal{S}}(P(\mathcal{K}))$. The resulting debiasing estimator is

$$\langle C(\mathcal{S}) \rangle = D(1) + \frac{\Delta D(k)}{p_k}. \tag{10}$$

Similar to the approach by Misso et al. [2022], we correlate estimates for $D(k)$ and $D(k+1)$ by using the same set of pixels. We uniformly sample k *distinct* secondary pixels to obtain a set of $k+1$ pixels $\mathcal{K}$. To reuse samples as much as possible, we loop over all k different ways of removing one *secondary* pixel from $\mathcal{K}$ and estimate $\langle D(k+1) - D(k) \rangle$ for each of the removed pixels as

$$\langle D(k+1) - D(k) \rangle = \frac{1}{k} \sum_{j \in \mathcal{K}, j \neq c} A_{\mathcal{S}}(P(\mathcal{K})) - A_{\mathcal{S}}(P(\mathcal{K} - \{j\})). \tag{11}$$

The remaining problem is to find a suitable p_k. Choosing p_k to minimize the $\mathbf{V}[\langle C(\mathcal{S}) \rangle]$ as was done by Misso et al. [2022] is not very helpful in our case since we rather want to minimize the work-variance $\mathbf{V}[\langle C(\mathcal{S})_k \rangle]\mathbf{C}[\langle C(\mathcal{S})_k \rangle]$ to account for both the variance and the cost. Minimizing the work-variance directly is challenging as it requires the knowledge of $C(\mathcal{S})$ itself. We, however, empirically found that having $p_k \propto k^{-2}$ produces reasonably low variance through numerical experiments. The average number of point evaluation $\sum_{k=1}^{|\mathcal{S}|} k p_k$ is a normalized partial sum of harmonic series that is known to grow only *logarithmically* to k, achieving a sublinear cost for each estimate of $P(\mathcal{S})$.

Efficient Interpolation Operator. While there are many different 1D interpolation operators, we found that simple linear interpolation works well. We experimented with high order polynomial interpolation but found that they usually increase variance as the interpolated contributions are not bounded by the minimum or maximum of $P(\mathcal{K})$. We thus approximate $P(\mathcal{S})$ by linearly interpolating weighted path contributions evaluated at two neighboring secondary pixels, and compute Eq. (7) using the interpolated values. For pixels j beyond the endpoints, we define $(A_{\mathcal{S}})_j$ by using the contribution of nearest endpoint. A naive implementation of Eq. (11) requires evaluating the interpolation operator k times, resulting in a overall $O(k|\mathcal{S}|)$ complexity. We can speed it up to $O(|\mathcal{S}|)$ by taking the advantage of the fact that the contribution of each j is only affected by the two neighboring pixels. For each pixel j, we first interpolate it by its neighbors, then we consider the alternative interpolation results by removing one of its neighbors and compute the difference. It only requires keeping track of two nearest neighbors in both directions. The total cost of computing Eq. (11) in this manner is $O(|\mathcal{S}|)$.

4.4 Generalization to 2D Shapes

Most concepts in previous sections such as shapes, MCMC weights, and debiasing estimators, generalize to 2D without modification. The only exception is the interpolation operator $A_{\mathcal{S}}$. We have to choose a scheme that allows evaluating Eq. (11) in linear time to prevent the computation cost exploding at large shapes. We explored a few interpolation methods and found out that simple 1D linear interpolation along a space filling curve (e.g. Hilbert curve) works well. For each shape $\mathcal{S}^i$, we run a space filling curve throughout $\mathcal{S}$ and order the pixels according to the curve, reducing the shape back into 1D. The secondary pixels are subsequently sampled and sorted in this order.

4.5 Spatially Varying Shape

Our method can support spatially varying shapes across pixel to further reduces variance in Eq. (10). Since the debiasing estimator works best when the difference between base estimator $D(1)$ and point estimator $P(\mathcal{S})$ is small, we would like to ideally include only those pixels that have similar contributions. We estimate pixel similarity based on auxiliary features such as albedo and geometry normals, similar to feature-preserving denoiser kernels [Dammertz et al. 2010; Rousselle et al. 2013]. To construct such shapes, we first assign a spatially uniform *base shape* $\mathcal{S}_B$ to every pixel in the image. We redefine $\mathcal{S}^i$ at each pixel i by selecting pixels $j \in \mathcal{S}_B^i$ such that

$$n_i \cdot n_j \ge \tau_n \wedge ||c_i - c_j||_\infty \le \tau_c, \tag{12}$$

where n_i and c_i are the geometry normal and albedo at i. We empirically set the thresholds for the differences τ_n and τ_c as $\tau_n = \sqrt{2}/2, \tau_c = 0.1$. Note that the shape Eq. (12) is deliberately constructed to be symmetric; if $j \in \mathcal{S}^i$, we also have $i \in \mathcal{S}^j$. It is because our shape splatting formulates secondary samples as MCMC mutations, and symmetric shapes ensure that all the secondary samples receive non-zero weights.

4.6 Decorrelating Shape Splats

Shape splatting results in lower numerical error by increasing the effective number of samples per pixel. However, it often introduces lower frequency noise due to inter-pixel correlation as also seen in

other path-reusing methods. We propose a simple technique to address this issue. We first inserts strides between pixels in the shape, shifting the noise to higher frequencies. We additionally randomly rotate the shapes in each sample, further decorrelating shape splats. This approach generally makes the correlation less perceptible in the image space, but shifting mapping to such randomized points increases per-pixel error by $\approx 10 \sim 15\%$, with large strides resulting in higher error but less correlation. One benefit is that a decorrelated image can usually be denoised effectively by a neural network denoiser such as OpenImageDenoise [Áfra 2019] in Fig. 3. Due to the flexibility of shape splatting and that we have direct control over the splatting location, it might be possible to combine our framework with blue-noise dithered sampling [Georgiev and Fajardo 2016] to achieve a more desirable error distribution over the image.

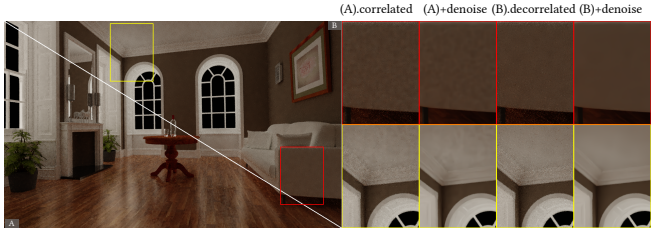

Fig. 3. We insert strides and randomly rotating the shape to greatly reduce low frequency artifacts and correlation between neighbor pixels. The decorrelated image can be further denoised by a machine learning denoiser without introducing outstanding artifacts.

4.7 Optional Mixing with Point Splats

Our debiasing estimator is not free from occasional artifacts or even negative-valued pixels when the base estimator differs significantly from the point estimator, due to outliers in the path contribution that cannot be easily detected via auxiliary features in Sec. 4.5. For example, when the primary sample is a high energy outlier, its overly high contribution is likely to spread to neighbor pixels to produce a group of outlier pixels, especially if there are not enough secondary samples to correct the bias. The artifact can also be produced if one of the secondary sample is an outlier.

We empirically found this issue is primarily relevant to shape splatting combined with vanilla path tracers, although shape splatting with other methods also benefit from point mixing. We alleviate this issue by combining the already-available point samples at each center pixel and secondary samples as they form two unbiased estimators that are free of such artifacts. Since these three estimators are correlated and use samples drawn from the *same* density, we cannot use MIS weights to weight the estimators. Instead, we compute per-pixel variance estimates for each estimator and weight them according to their inverse variance. Unfortunately, it introduces small amount of bias manifests as energy loss around high variance regions due to the variance estimates being correlated with the estimators. Nevertheless, we found the trade-off to be worthwhile as it is generally effective in removing artifacts. Investigating an unbiased mixing scheme could be an interesting future work.

5 RESULTS

We implemented our method on a CPU/GPU rendering framework powered by the Rust frontend [Tong et al. 2023] of LuisaCompute [Zheng et al. 2022]. We built it upon a mega-kernel path tracer, unidirectional PSSMLT [Kelemen et al. 2002], and offline ReSTIR [Lin et al. 2022][1]. Our method can be easily integrated into existing rendering systems, requiring no major change to the path tracer, especially when the rendering system has already implemented hybrid shift mapping (e.g., ReSTIR). We also propose a simple and efficient implementation of our method on GPUs. Since each pixel uses a potentially different number of debiasing samples, naively implementing our method results in poor performance due to thread divergence. We propose to implement shape splatting in a wavefront manner while keeping the mega-kernel path tracer: we launch separate kernels to sample primary paths, the secondary paths, and compute telescoping sum to guarantees that all threads have equal work loads. Compared to our CPU implementation, our GPU implementation only requires tens of lines of additional code to implement this wavefront scheduling logic. The memory overhead is negligible as it only requires recording the correspondence between secondary and primary samples. Our program spends around 50% of time tracing primary paths, 40% on secondary paths, and 10% on computing the telescoping sum and splatting contribution. We implemented hybrid shift mapping [Lin et al. 2022] due to its simplicity and effectiveness, where we evaluate the Jacobian and sample distribution $p(\cdot)$ under primary sample space parameterization. The optimal shape size depends on the image resolution as well as shift mapping efficiency. Our method works well for shape sizes ranging from $20 \sim 80$ pixels. Using shapes larger than 80 pixels are possible but subjected to increased memory overhead during splatting. Finding the optimal shape for a given scene and image resolution is left for future work.

We measure the performance of our shape splatting on top of PT and offline ReSTIR PT on an NVIDIA RTX 3070Ti GPU, where we additionally compared with Bekaert-style path reusing (BPR). For ReSTIR PT, we sample 32 initial candidates and perform 3 rounds of spatial reuse with 6 neighbors each round in a 10 pixel radius. We offered two variants for our method combined with ReSTIR: the RIS PT+Ours variant replace all 3 spatial reuse rounds with 8 rounds of shape splatting using randomized 61 pixel shapes; the ReSTIR PT+Ours variant replace the last spatial reuse in ReSTIR with 8 rounds of shape splatting. Shape splatting with PSSMLT are measured on an Intel i9-13900K 24C/32T processor. We tested our method on a variety of scenes in equal time settings. Fig. 1 and Fig. 10 show some closeups. [2] Since offline ReSTIR PT is mainly for computing indirect lighting, we compare all methods on *indirect lighting only*. However, our method is also able to handle direct lighting efficiently. We show error plots of our method against their point splatting baselines in MAPE metrics in Fig. 8 and Fig. 9.

[1]The code is available at https://github.com/splatting-shapes/Efficient-Image-Shape-Splatting

[2] *LIVING ROOM* by courtesy of Jay-Artists under CC-BY 3.0; *FIREPLACE ROOM* and *BREAKFAST ROOM* by courtesy of Wig42 under CC-BY 3.0; *SALLE DE BAIN* by courtesy of nacimus under CC-BY 3.0; *SAN MIGUEL* by courtesy of Guillermo M. Leal Llaguno under CC-BY 3.0; *BISTRO* by courtesy of Amazon Lumberyard under CC-BY 4.0; *CLASSROOM* by courtesy of Christophe Seux under CC0

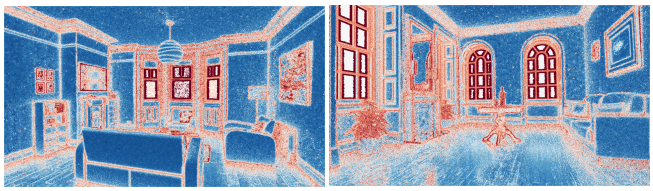

Fig. 4. Visualizing the weights when mixing with point samples. Blue corresponds to a higher weight for shape splatting than points. Our method works best when the geometry and lighting is smooth, but it can also handle non-smooth shading such as image textures.

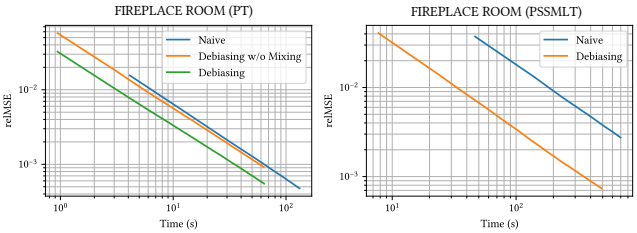

Fig. 5. We compared our debiasing estimator with naive shape splatting under equal time. In PSSMLT+Ours (right), our debiasing estimator outperforms naive shape splatting by a large margin. In PT+Ours (left), while debiasing without mixing with point samples produces higher relMSE due to the presence of outliers, mixing with point samples effectively suppresses outliers and has lower variance than naive shape splatting.

The results show that our method consistently outperforms point splatting in all test scenes under equal time. Our debiasing and secondary sample generation by shift mapping allows us to reduce an actual cost of shape splatting equivalent to *only two point samples* on average. With the help of spatially varying kernels, our shape splatting can adapt to textured materials and sharp geometry edges. Fig. 8 demonstrates how our shape splatting can accelerate a vanilla path tracer and existing path reusing methods such as ReSTIR. The improvements on top of the path reusing methods show that our shape splatting is not simply yet another path reuse method. For example, even RIS PT+Ours is already almost as effective as ReSTIR on many scenes. Note that one does not need to choose between RIS PT+Ours and ReSTIR PT as that our method and ReSTIR are not exclusive to each other. One can achieve better performance with the combined ReSTIR PT+Ours variant. RIS/ReSTIR PT+Ours is also free from color noises that usually present in ReSTIR PT (Fig. 1). Our decorrelation approach is generally effective at reducing artifacts except for very challenging scenes or when the base estimator itself exhibits strong inter-pixels correlation. (e.g. *DIFFICULT LIVING ROOM* scene in Fig. 10)

5.1 Ablation Study

We conduct an ablation study to investigate the effectiveness of various components of our method in Fig. 6. We focus on analyzing shape splatting with PT in this section.

Mixing with point samples. Fig. 6 col. **B** is a result of using only primary and secondary *point* samples during shape splatting. It is quite noisy due to its point splatting nature. Col **D**. instead uses only shape samples. While shape splatting reduces noise, it produces outliers around texture discontinuities and geometry edges. Mixing points and shapes (col **F**) using weighted inverse variance removes outliers effectively while preserving the low variance of the shape estimator. Fig. 4 visualizes mixing weights on a few scenes. Shape splatting receives larger weights in most cases. In regions where high frequency geometry dominates, our method offer less improvement over point splatting as the contribution of shape splatting is negated by the variance from debiasing.

Effect of debiasing. To demonstrate that debiasing is indeed necessary, in col.**E** Fig. 6, we only compute the telescoping sum up to $k = 3$ in Eq. (10) so that the bias is not completely removed. While it has lower error at low sample count, it is neither unbiased nor consistent. Notice the obvious blurring of texture and geometry, even when we already use spatially varying shapes described in Sec. 4.5 to group similar pixels.

Comparison between pairwise MIS and MCMC weights. To evaluate MCMC weights in Eq. (7), we compared them against pairwise MIS [Bitterli 2021] in col. **C** of Fig. 6. We found that MIS weights tend to produce higher variance and more outliers than our MCMC weights.

Comparison of PMFs for Debiasing. We have tested different choices of the PMF in Eq. (10) with the form of $p_k \propto k^{-m}$. If we ignore the cost of the estimator and chose the PMF according to variance only, $p_k = k^{-1}$ has lowest variance among them. However, once we consider the overall efficiency, our choice of $p_k = k^{-2}$ performs the best.

Debiased and naive shape splatting. Our shape estimator introduces additional noise due to debiasing. This raises the question of whether the advantages of splatting shapes outweigh this noise. In Fig. 5, we compared against naive shape splatting where all pixels in the shape is estimated by point sampling from scratch. Our debiasing estimator outperforms the naive approach even without mixing. Once we mix with point samples, our debiasing estimators significantly outperforms naive shape splatting. To provide further insights, we provide closeups in Fig. 7. Under equal shape sample, our debiasing estimator performs equally well as naive splatting in smooth regions but with significantly lower computational cost, while naive splatting exhibits lower variance around geometry discontinuities. In equal time comparisons our debiasing estimator already outperforms in smooth region due to reduced correlation artifacts due to higher sample count, except for the presence of outliers. Mixing with point samples effectively eliminates these outliers and results in a lower mean image error.

6 CONCLUSION

We proposed a framework for efficient shape splatting in image space by leveraging MCMC mutations and debiasing. While our framework can improve various point-sample counterparts already, it has a few aspects that can be further studied.

Firstly, shape splatting may not work efficiently if the underlying rendering algorithm has a large variance (i.e., firefly noise). In this case, a sampled shape might splat outlier samples to other pixels in

the shape, resulting in a visual artifact. Mixing with point samples helps but may result in energy loss due to the biased nature of this mixing. Utilizing combiner methods developed for denoising [Back et al. 2020, 2022; Gu et al. 2022; Zheng et al. 2021] to mix point splats and shape splats could be helpful.

Secondly, our interpolation scheme only interpolates weighted path contribution, but it potentially enables a wide range of interpolation schemes. For example, one could decompose path contribution into BSDF and irradiance terms and interpolate them separately. Investigating a better interpolation scheme could improve performance in the presence of high frequency geometries and textures.

Lastly, applications of our framework to any other problems involving many correlated integrals would be interesting to explore. For example, integrating our framework with Monte Carlo estimators of partial differential equations [Rioux-Lavoie et al. 2022; Sawhney and Crane 2020; Sugimoto et al. 2023] is likely fruitful because one would usually want to run such an estimator for consecutive points in space just like pixels in images.

ACKNOWLEDGMENTS

We would like to thank all anonymous reviewers for their constructive feedback; Eric Heitz for participating in early discussion; Qiqin (Maxwell) Fang and Weijie Zhou for discussion on the debiasing idea; Shaokun Zheng, Jiawei Huang, and Wenyou Wang for proofreading. This research was funded NSERC Discovery Grants (RGPIN-2020-03918).

REFERENCES

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Trans. Graph.* 39, 6 (2020), 242:1–242:12. https://doi.org/10.1145/3414685.3417847

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2022. Self-Supervised Post-Correction for Monte Carlo Denoising. In *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann (Eds.). ACM, 18:1–18:8. https://doi.org/10.1145/3528233.3530730

Rasmus Barringer, Carl Johan Gribel, and Tomas Akenine-Möller. 2012. High-quality curve rendering using line sampled visibility. *ACM Trans. Graph.* 31, 6 (2012), 162:1–162:10. https://doi.org/10.1145/2366145.2366181

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain path reusing. *ACM Trans. Graph.* 36, 6 (2017), 229:1–229:9. https://doi.org/10.1145/3130800.3130886

Philippe Bekaert, Mateu Sbert, and John H. Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Proceedings of the 13th Eurographics Workshop on Rendering Techniques, Pisa, Italy, June 26-28, 2002 (ACM International Conference Proceeding Series, Vol. 28)*, Simon Gibson and Paul E. Debevec (Eds.). Eurographics Association, 125–134. https://doi.org/10.2312/EGWR/EGWR02/125-134

Benedikt Bitterli. 2021. *Correlations and Reuse for Fast and Accurate Physically Based Light Transport.* Ph. D. Dissertation. Dartmouth College.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron E. Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4 (2020), 148. https://doi.org/10.1145/3386569.3392481

David Cline and Parris Egbert. 2005. A practical introduction to metropolis light transport. *Brigham Young University* (2005).

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. 2010. Edge-avoiding À-Trous wavelet transform for fast global illumination filtering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics 2010, Saarbrücken, Germany, June 25-27, 2010*, Justin Hensley, Philipp Slusallek, David K. McAllister, and Christiaan P. Gribble (Eds.). Eurographics Association, 67–75. https://doi.org/10.2312/EGGH/HPG10/067-075

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-noise dithered sampling. In *ACM SIGGRAPH 2016 Talks*. 1–1.

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Carl Johan Gribel, Rasmus Barringer, and Tomas Akenine-Möller. 2011. High-quality spatio-temporal rendering using semi-analytical visibility. *ACM Trans. Graph.* 30, 4 (2011), 54. https://doi.org/10.1145/2010324.1964949

Carl Johan Gribel, Michael C. Doggett, and Tomas Akenine-Möller. 2010. Analytical motion blur rasterization with compression. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on High Performance Graphics 2010, Saarbrücken, Germany, June 25-27, 2010*, Justin Hensley, Philipp Slusallek, David K. McAllister, and Christiaan P. Gribble (Eds.). Eurographics Association, 163–172. https://doi.org/10.2312/EGGH/HPG10/163-172

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-domain volumetric photon density estimation. *ACM Trans. Graph.* 37, 4 (2018), 82. https://doi.org/10.1145/3197517.3201363

Jeongmin Gu, José Antonio Iglesias Guitián, and Bochang Moon. 2022. Neural James-Stein Combiner for Unbiased and Biased Renderings. *ACM Trans. Graph.* 41, 6 (2022), 262:1–262:14. https://doi.org/10.1145/3550454.3555496

W. K. Hastings. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* 57, 1 (1970), 97–109. http://www.jstor.org/stable/2334940

Thouis R. Jones and Ronald N. Perry. 2000. Antialiasing with Line Samples. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000, Brno, Czech Republic, June 26-28, 2000 (Eurographics)*, Bernard Peroche and Holly E. Rushmeier (Eds.). Springer, 197–206. https://doi.org/10.1007/978-3-7091-6303-0_18

James T. Kajiya. 1986. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986, Dallas, Texas, USA, August 18-22, 1986*, David C. Evans and Russell J. Athay (Eds.). ACM, 143–150. https://doi.org/10.1145/15922.15902

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. https://doi.org/10.1111/1467-8659.T01-1-00703

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain path tracing. *ACM Trans. Graph.* 34, 4 (2015), 123:1–123:13. https://doi.org/10.1145/2766997

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph.* 32, 4 (2013), 95:1–95:12. https://doi.org/10.1145/2461912.2461943

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: foundations of ReSTIR. *ACM Trans. Graph.* 41, 4 (2022), 75:1–75:23. https://doi.org/10.1145/3528223.3530158

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing. In *26th Eurographics Symposium on Rendering, Rendering - Experimental Ideas & Implementations, EGSR 2015, EI&I Track, Darmstadt, Germany, June 23-26, 2015*, Jaakko Lehtinen and Derek Nowrouzezahrai (Eds.). Eurographics Association, 65–74. https://doi.org/10.2312/SRE.20151168

Zackary Misso, Benedikt Bitterli, Iliyan Georgiev, and Wojciech Jarosz. 2022. Unbiased and consistent rendering using biased estimators. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022). https://doi.org/10/gqjn66

Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H. Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2022. A Monte Carlo Method for Fluid Simulation. *ACM Trans. Graph.* 41, 6 (2022), 240:1–240:16. https://doi.org/10.1145/3550454.3555450

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising using Feature and Color Information. *Comput. Graph. Forum* 32, 7 (2013), 121–130. https://doi.org/10.1111/CGF.12219

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo geometry processing: a grid-free approach to PDE-based methods on volumetric domains. *ACM Trans. Graph.* 39, 4, Article 123 (aug 2020), 18 pages. https://doi.org/10.1145/3386569.3392374

Peter Shirley and Chris Wyman. 2017. Generating stratified random lines in a square. *Journal of Computer Graphics Techniques (JCGT)* 6, 2 (30 June 2017), 49–54. http://jcgt.org/published/0006/02/03/

Gurprit Singh, Bailey Miller, and Wojciech Jarosz. 2017. Variance and Convergence Analysis of Monte Carlo Line and Segment Sampling. *Comput. Graph. Forum* 36, 4 (2017), 79–89. https://doi.org/10.1111/CGF.13226

Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Trans. Graph.* 42, 4, Article 81 (jul 2023), 16 pages. https://doi.org/10.1145/3592109

Weilun Sun, Xin Sun, Nathan A. Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-Domain Vertex Connection and Merging. In *28th Eurographics Symposium on Rendering, Rendering - Experimental Ideas & Implementations, EGSR 2017, EI&I Track, Helsinki, Finland, 19-21 June 2017*, Matthias Zwicker and Pedro V. Sander (Eds.). Eurographics Association, 83–92. https://doi.org/10.2312/SRE.20171197

Xin Sun, Kun Zhou, Jie Guo, Guofu Xie, Jingui Pan, Wencheng Wang, and Baining Guo. 2013. Line segment sampling with blue-noise properties. *ACM Trans. Graph.* 32, 4 (2013), 127:1–127:14. https://doi.org/10.1145/2461912.2462023

Xiaochun Tong, Hsueh-Ti Derek Liu, Yotam I. Gingold, and Alec Jacobson. 2023. Differentiable Heightfield Path Tracing with Accelerated Discontinuities. In *ACM*

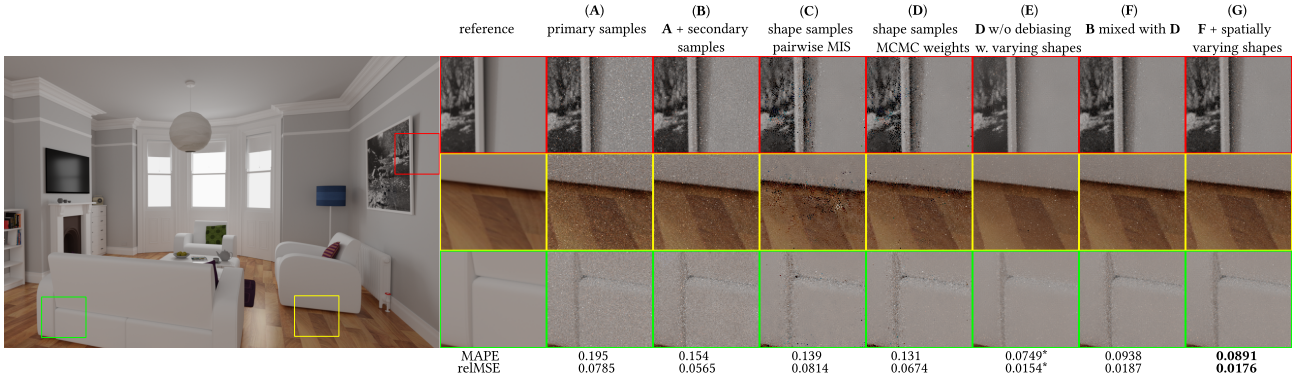

Fig. 6. We demonstrate the performance of different components of our method. Note while **E** has lower error, it is inconsistent and has blurring artifact.

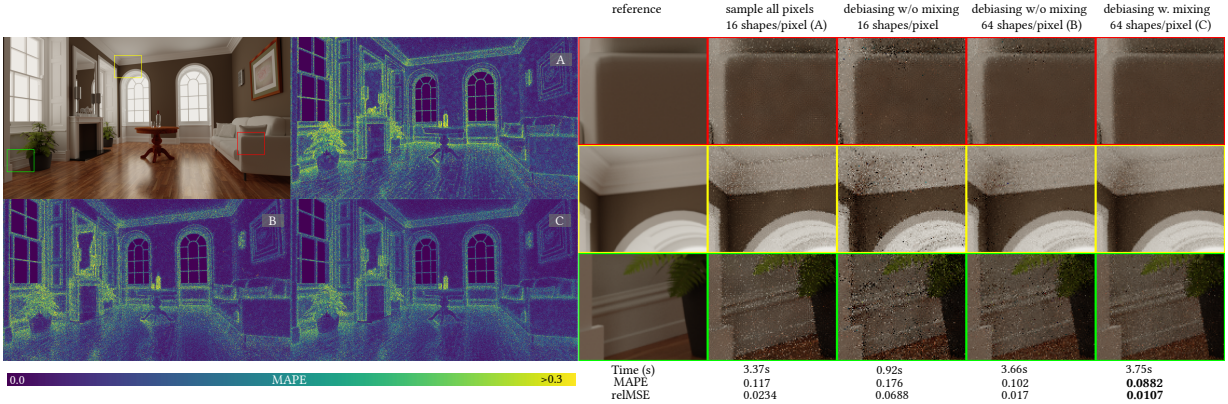

Fig. 7. Our debiasing estimator outperforms naive shape splatting, mixing with point samples improves efficiency further by removing outliers.

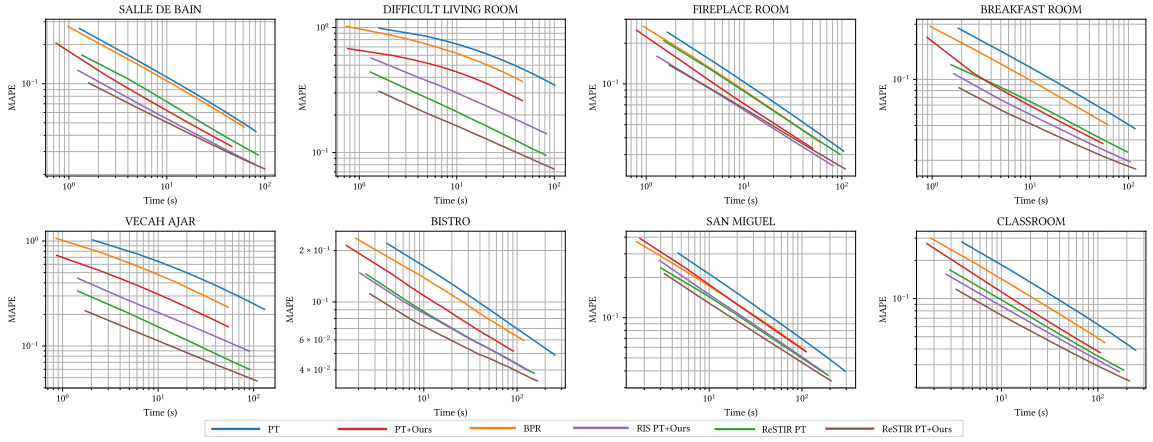

Fig. 8. Error plots of shape splatting vs point splatting baselines on GPU.

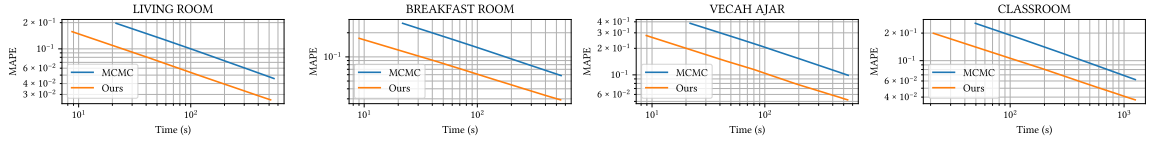

Fig. 9. Error plots of shape vs point splatting with PSSMLT

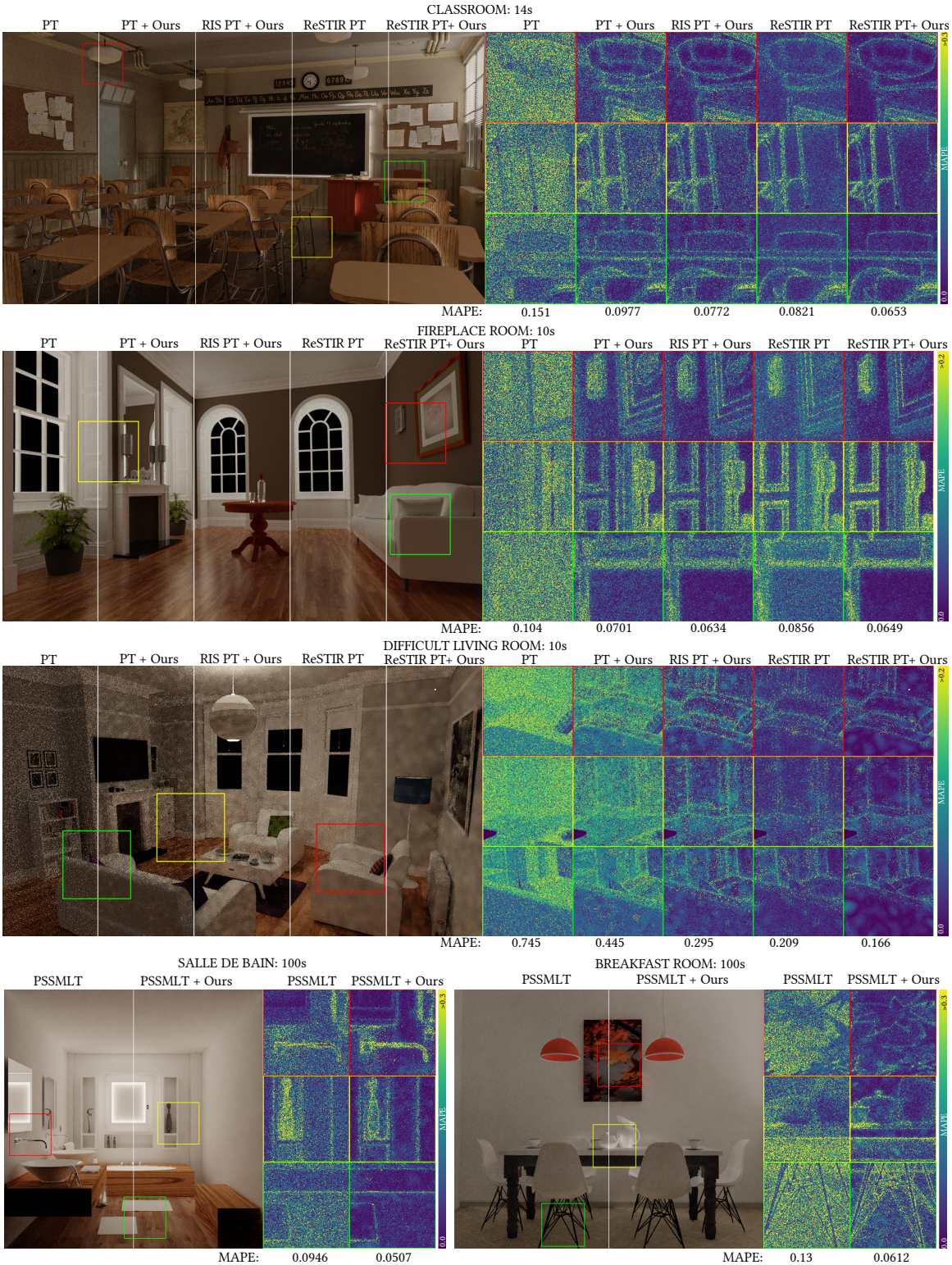

Fig. 10. We integrated our method on top of PT, PSSMLT and ReSTIR PT. Our method consistently outperforms point splatting in a variety of scenes.

SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023, Erik Brunvand, Alla Sheffer, and Michael Wimmer (Eds.). ACM, 19:1–19:9. https://doi.org/10.1145/3588432.3591530

Stanley Tzeng, Anjul Patney, Andrew A. Davidson, Mohamed S. Ebeida, Scott A. Mitchell, and John D. Owens. 2012. High-Quality Parallel Depth-of-Field Using Line Samples. In *Proceedings of the EUROGRAPHICS Conference on High Performance Graphics 2012, Paris, France, June 25-27, 2012*, Carsten Dachsbacher, Jacob Munkberg, and Jacopo Pantaleoni (Eds.). Eurographics Association, 23–31. https://doi.org/10.2312/EGGH/HPG12/023-031

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation.* Ph. D. Dissertation. Stanford University, USA. https://searchworks.stanford.edu/view/3911108

Eric Veach and Leonidas J. Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, Los Angeles, CA, USA, August 6-11, 1995*, Susan G. Mair and Robert Cook (Eds.). ACM, 419–428. https://dl.acm.org/citation.cfm?id=218498

Eric Veach and Leonidas J. Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997, Los Angeles, CA, USA, August 3-8, 1997*, G. Scott Owen, Turner Whitted, and Barbara Mones-Hattal (Eds.). ACM, 65–76. https://doi.org/10.1145/258734.258775

Shaokun Zheng, Fengshi Zheng, Kun Xu, and Ling-Qi Yan. 2021. Ensemble denoising for Monte Carlo renderings. *ACM Trans. Graph.* 40, 6 (2021), 274:1–274:17. https://doi.org/10.1145/3478513.3480510

Shaokun Zheng, Zhiqian Zhou, Xin Chen, Difei Yan, Chuyan Zhang, Yuefeng Geng, Yan Gu, and Kun Xu. 2022. LuisaRender: A High-Performance Rendering Framework with Layered and Unified Interfaces on Stream Architectures. *ACM Trans. Graph.* 41, 6 (2022), 232:1–232:19. https://doi.org/10.1145/3550454.3555463

Attila T Áfra. 2019. OpenImageDenoise. https://www.openimagedenoise.org/