Supplemental: Marginal Multiple Importance Sampling

REX WEST, The University of Tokyo, Japan
ILIYAN GEORGIEV, Autodesk, United Kingdom
TOSHIYA HACHISUKA, University of Waterloo, Canada

In this supplemental document we provide the following: a proof of unbiasedness of the marginal MIS (MMIS) estimator (see Section 1), a version of the MMIS estimator with arbitrary weighting functions (see Section 2), a detailed look at how to implement a performant marginal path sampling based multi-vertex path filtering (MVPF) in practice (see Section 3), and additional rendering results for multi-vertex path filtering and multi-vertex photon filtering (see Section 4).

ACM Reference Format:
Rex West, Iliyan Georgiev, and Toshiya Hachisuka. 2022. Supplemental: Marginal Multiple Importance Sampling. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3550469.3555388

1 UNBIASEDNESS OF THE MARGINAL MIS ESTIMATOR

We show the unbiasedness of our MMIS estimator by writing out its expected value over all random variables, i.e., the $N = \sum_{i=1}^{T} n_i$ independently sampled pairs $(t_{i,j}, x_{i,j}) \sim p_i(t_{i,j}, x_{i,j}) = p_i(t_{i,j}) p_i(x_{i,j}|t_{i,j})$ where i is the index to each technique space, and j is the index to each sample per technique space:

$$\mathrm{E}[\langle I\rangle_{\mathrm{Mar}}] = \mathrm{E}[\langle I\rangle_{\mathrm{Mar}}]_{(t_{1,1},x_{1,1}),\ldots,(t_{T,n_T},x_{T,n_T})} \tag{1a}$$

$$= \mathrm{E}\Big[\mathrm{E}[\langle I\rangle_{\mathrm{Mar}}]_{x_{1,1},\ldots,x_{T,n_T}}\Big]_{t_{1,1},\ldots,t_{T,n_T}} \tag{1b}$$

$$= \mathrm{E}\left[\mathrm{E}\left[\sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{f(x_{i,j})}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x_{i,j}|t_{i',j'})}\right]_{x_{1,1},\ldots,x_{T,n_T}}\right]_{t_{1,1},\ldots,t_{T,n_T}} \tag{1c}$$

$$= \mathrm{E}\left[\sum_{i=1}^{T}\sum_{j=1}^{n_i} \int_{\mathcal{X}} \frac{f(x)}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x|t_{i',j'})} p_i(x|t_{i,j})\,\mathrm{d}x\right]_{t_{1,1},\ldots,t_{T,n_T}} \tag{1d}$$

$$= \mathrm{E}\left[\int_{\mathcal{X}} \underbrace{\sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{p_i(x|t_{i,j})}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x|t_{i',j'})}}_{=1} f(x)\,\mathrm{d}x\right]_{t_{1,1},\ldots,t_{T,n_T}} \tag{1e}$$

$$= \mathrm{E}\left[\underbrace{\int_{\mathcal{X}} f(x)\,\mathrm{d}x}_{=I}\right]_{t_{1,1},\ldots,t_{T,n_T}} = I\,, \tag{1f}$$

where for each x in $\mathcal{X}$, where $f(x) > 0$, at least one sampled technique such that $p_i(x|t_{i,j}) > 0$, and the weighting term $\frac{p_i(x|t_{i,j})}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x|t_{i',j'})}$ sums to 1 over the N technique-sample pairs.

2 MARGINAL MIS AND APPROXIMATE WEIGHTS

Although MMIS is derived as an approximation of the balance heuristic for multi-sample MIS, its unbiasedness still holds for any arbitrary weighting function that similarly sum to 1 over the $\sum_{i=1}^{T} n_i$ independently drawn technique-sample pairs:

$$\langle I\rangle_{\mathrm{Mar^*}} = \sum_{i=1}^{T}\sum_{j=1}^{n_i} w_i(t_{i,j}, x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j}|t_{i,j})}\,, \tag{2}$$

where for each sample $x \in \mathcal{X}$, when $f(x) > 0$, $\sum_{i=1}^{T}\sum_{j=1}^{n_i} w_i(t_{i,j}, x)$ sums to 1.

Replacing the weighting function w_i in Eq. (2) with the balance heuristic,

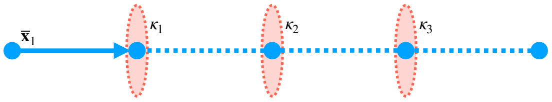
$$w_i(t,x) = \frac{p_i(x|t)}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x|t_{i',j'})}\,, \tag{3}$$

yields the estimator in the main paper.

3 MULTI-VERTEX PATH FILTERING IMPLEMENTATION DETAILS

Directly implementing a marginal path sampling (MPS) estimator for multi-vertex path filtering (MVPF) can result in a fairly inefficient implementation. In this section we will look at how to apply MPS theory to the MVPF problem, and then show how we can take advantage of the structure of the formulation to implement a performant algorithm.

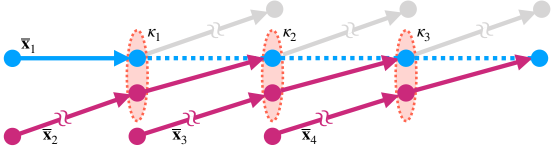
The goal of path filtering [Keller et al. 2014; West et al. 2020] is to refine the outgoing radiance of a vertex by reusing the incoming radiance of other nearby vertices (generally from different paths). Multi-vertex path filtering extends this idea from one vertex along a path, to multiple vertices (e.g. intermediate vertices—all but the first and last vertex of a path).



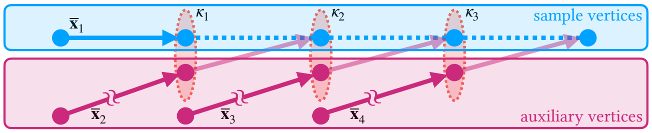
The filtering performed at each vertex is often implemented as a range query or clustering operation. This effectively selects a finite

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

set of prefixes and their next vertex who's incoming radiance we would like to reuse.



As each prefix conditions the sampling of a next vertex, choosing a prefix at *every* filtering kernel along a path conditions the sampling of a *complete path*.



The collection of prefixes forms a set of auxiliary vertices $\bar{\mathbf{t}}$, giving us a conditional technique in marginal path sampling. This conditional technique has a sampling PDF that is the product of sample vertices $\mathbf{x}$ given the prefixes $\bar{\mathbf{y}}$ that condition their sampling,

$$p(\bar{\mathbf{x}}|\bar{\mathbf{t}}) = \prod_{k'=2}^{k-1} p(\mathbf{x}_{k'}|\bar{\mathbf{y}}_{k'}) \,. \tag{4}$$

where k is the number of vertices in the path sample $\bar{\mathbf{x}}$. This conditional technique belongs to a marginal technique over the space of possible vertex values for the selected prefixes.

The set of conditional techniques we will use in MVPF is then all of the ways we can sample a given path $\bar{\mathbf{x}}$ — the choices of prefix vertices that fall within the support of the filtering kernels along the path $\bar{\mathbf{x}}$. If we apply filtering at the intermediate $k-2$ vertices, there are $T = \prod_{k'=2}^{k-1} N(\mathbf{P}, \kappa_{k'})$ conditional path sampling techniques, where $\mathbf{P}$ is the set of traced paths and $N(\mathbf{P}, \kappa_{k'})$ counts the number of prefixes that fall within the support of the filtering kernel $\kappa_{k'}$. This results in a number of conditional techniques exponential in k.

Computing the denominator of the marginal path sampling weights would generally require computing the PDF of $\bar{\mathbf{x}}$ for each of the exponential number of conditional techniques,

$$\sum_{i'=1}^{T} \sum_{j'=1}^{n_{i'}} p_{i'}(\bar{\mathbf{x}}_{i,j}|\bar{\mathbf{t}}_{i',j'}) = \sum_{i'=1}^{T} \sum_{j'=1}^{n_{i'}} \underbrace{\prod_{k'=2}^{k-1} p_{i'}(\mathbf{x}_{i,j,k'}|\bar{\mathbf{y}}_{i',j',k'})}_{p_{i'}(\bar{\mathbf{x}}_{i,j}|\bar{\mathbf{t}}_{i',j'})} \,. \tag{5}$$

where, $\mathbf{x}_{i,j,k'}$ is the k'^{th} vertex from the sensor on the j^{th} path sample from the i^{th} marginal technique, and $\bar{\mathbf{y}}_{i',j',k'}$ is the prefix used to compute the conditional PDF of the sample vertex $\mathbf{x}_{i,j,k'}$. The index i iterates over the permutations of prefix choices at each kernel, and there by intrinsically parameterizes a choice of prefix at each kernel.

However, note that we are considering all of the ways to sample a path $\bar{\mathbf{x}}$. Here this is all of the possible *permutations* of choosing one prefix at each filtering kernel. As such, the conditional PDF of each sample vertex $\mathbf{x}$ given their prefix $\bar{\mathbf{y}}$ is reused across many PDF terms $p_{i'}(\bar{\mathbf{x}}_{i,j}|\bar{\mathbf{t}}_{i',j'})$ in the denominator of the marginal path sampling estimator. With some reorganization of terms, changing the *sum-of-products* to a *product-of-sums*,

$$\sum_{i'=1}^{T} \sum_{j'=1}^{n_{i'}} p_{i'}(\bar{\mathbf{x}}_{i,j}|\bar{\mathbf{t}}_{i',j'}) = \prod_{k'=1}^{k} \underbrace{\sum_{m=1}^{N(\mathbf{P},\kappa_{k'})} p_m(\mathbf{x}_{i,j,k'}|\bar{\mathbf{y}}_{m,k'}) \,,}_{\text{filter local MIS}} \tag{6}$$

we can reduce the exponential computational complexity to linear in k where $\bar{\mathbf{y}}_{m,k'}$ is the m^{th} prefix that falls within the filter support at the k'^{th} vertex of the sample path. This reorganization of terms is analogous to locally performing MIS over each filtering kernel and taking their product.

Iterative multi-vertex path filtering algorithm. In practice we are filtering not just a single path, but many paths (e.g. one path per pixel). The local MIS weights of each sample vertex are used across every constructed path that contains that sample vertex. This gives us the opportunity to further amortize path weighting overhead by computing each local MIS weight once, storing it, and iteratively accumulating and propagating the local MIS weights. The resulting algorithm (see Algorithm 1) is analogous to the iterative path filtering of Deng et al. [2021].

Algorithm 1: Multi-vertex Path Filtering

```
Input: Number of filtering iterations $K$
for $pixel \in image$ do
    $path$ = tracePath($pixel$)
    $vertices.add(path.vertices)$
$clusters$ = cluster($vertices$)
for $c \in clusters$ do
    for $vertex \in c$ do
        $vertex.weight$ = computeLocalMIS($vertex, c$)
        $vertex.incRad$ = suffixOutRad($vertex.suffix$)
for $i \in [1, K]$ do
    for $c \in clusters$ do
        for $vertex \in c$ do
            $vertex.outRad$ = weightAndSumIncRad($vertex, c$)
    for $c \in clusters$ do
        for $vertex \in c$ do
            $vertex.incRad = vertex.nextVertex.outRad$
for $pixel \in image$ do
    $pixelVert$ = getFirstPathVertex($vertices, pixel$)
    $pixel.accumulate(pixelVert.incRad)$
```

By initially setting the incoming radiance of each vertex to the unbiased outgoing radiance of their suffix, and filtering all clusters some number of iterations K, we will have effectively performed multi-vertex path filtering at the first K intermediate vertices of every path.

Initial incoming radiance estimate. Some care needs to be taken when choosing an initial incoming radiance value used for each vertex. Some choices, such as setting the initial value to zero or direct light, result in bias in the form of energy loss. We chose to initialize the incoming radiance of each vertex to the unbiased outgoing radiance of their suffix. This allows us to perform an arbitrary number of filtering iterations and still guarantee the estimator samples paths of every length.

Next Event Estimation. In next event estimation (NEE) path tracing the last vertex of a path can be sampled by either BSDF (i.e. continuation) sampling or light sampling. This can greatly reduce estimation variance compared to purely unidirectional path sampling. We can incorporate NEE into the MVPF implementation discussed above by modifying the path sampling process and local MIS weights. During path sampling we additionally sample and store a light vertex for each vertex of a path, such that each path vertex now has a reference to its continuation vertex and a light vertex. During cluster-local filtering we then consider the incoming radiance from both the continuation vertices and the associated light vertices. This gives us an updated local MIS weight denominator for vertices that lie on a light source,

$$\sum_{m=1}^{N(\mathbf{P},\kappa_{k'})} p_m(\mathbf{x}_{k'}|\overline{\mathbf{y}}_{m,k'}) + N_L(\mathbf{P},\kappa_{k'})p_L(\mathbf{x}_{k'}), \qquad (7)$$

where $\mathbf{x}_{k'}$ is the k'^{th} vertex of a path sample, $N_L(\mathbf{P},\kappa_{k'})$ counts the number of light vertices for the k'^{th} filter kernel, and $p_L(\mathbf{x}_{k'})$ is the unconditional light sampling PDF of the vertex $\mathbf{x}_{k'}$.

Greedy range query clustering. For the results in the main paper and this supplemental we use a range query-based clustering (see Algorithm 2) that has several nice properties. The resulting clusters are guaranteed to have a radius no larger than the range query radius parameter. This results in a controllable amount of bias proportional to the radius. Further, the radius parameter can be progressively reduced to decrease the bias of each successive MVPF rendering pass. Further investigation may show that such progressive radius reduction results in a consistent algorithm.

Algorithm 2: Range query-based clustering

```
Input: Set of vertices from all paths $V$
Input: Range query radius parameter $r$
for $vertex \in V$ do
    if $vertex.clusterID \neq null$ then
        continue
    $id$ = nextClusterID()
    $vertex.clusterID = id$
    $nearby$ = rangeQuery($V, vertex, r$)
    for $nearVert \in nearby$ do
        if $nearVert.clusterID == null$ then
            $nearVert.clusterID = id$
```

4 ADDITIONAL RESULTS

In Fig. 1 we provide an equal-time comparison of multi-vertex path filtering (MVPF) and multi-vertex photon filtering (MVPhF) against four baseline methods: path tracing (PT), bidirectional path tracing (BDPT), single vertex path filtering (PF), and photon mapping (PM), on four additional scenes that cover various real-world lighting scenarios. It is particularly worth noting that for some scenes photon filtering (MVPhF) may not provide enough benefit over normal photon mapping to overcome the additional bias incurred when filtering.

ACKNOWLEDGMENTS

We express our gratitude to the following Blend Swap users: *Nacimus Ait Cherif* for the bathroom scene in Fig. 1, *Wig42* for the grey & white room in Fig. 1, *nulus* for the ceiling lamp used in rows 2 and 3 of Fig. 1, and *chams* for the alarm clock in Fig. 1.

REFERENCES

Xi Deng, Miloš Hašan, Nathan Carr, Zexiang Xu, and Steve Marschner. 2021. Path Graphs: Iterative Path Space Filtering. *ACM Trans. Graph.* 40, 6, Article 276 (dec 2021), 15 pages. https://doi.org/10.1145/3478513.3480547

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2014. Path Space Filtering *(SIGGRAPH ’14)*. ACM, 68:1–68:1. https://doi.org/10/gfz6mr

Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. 2020. Continuous Multiple Importance Sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392436

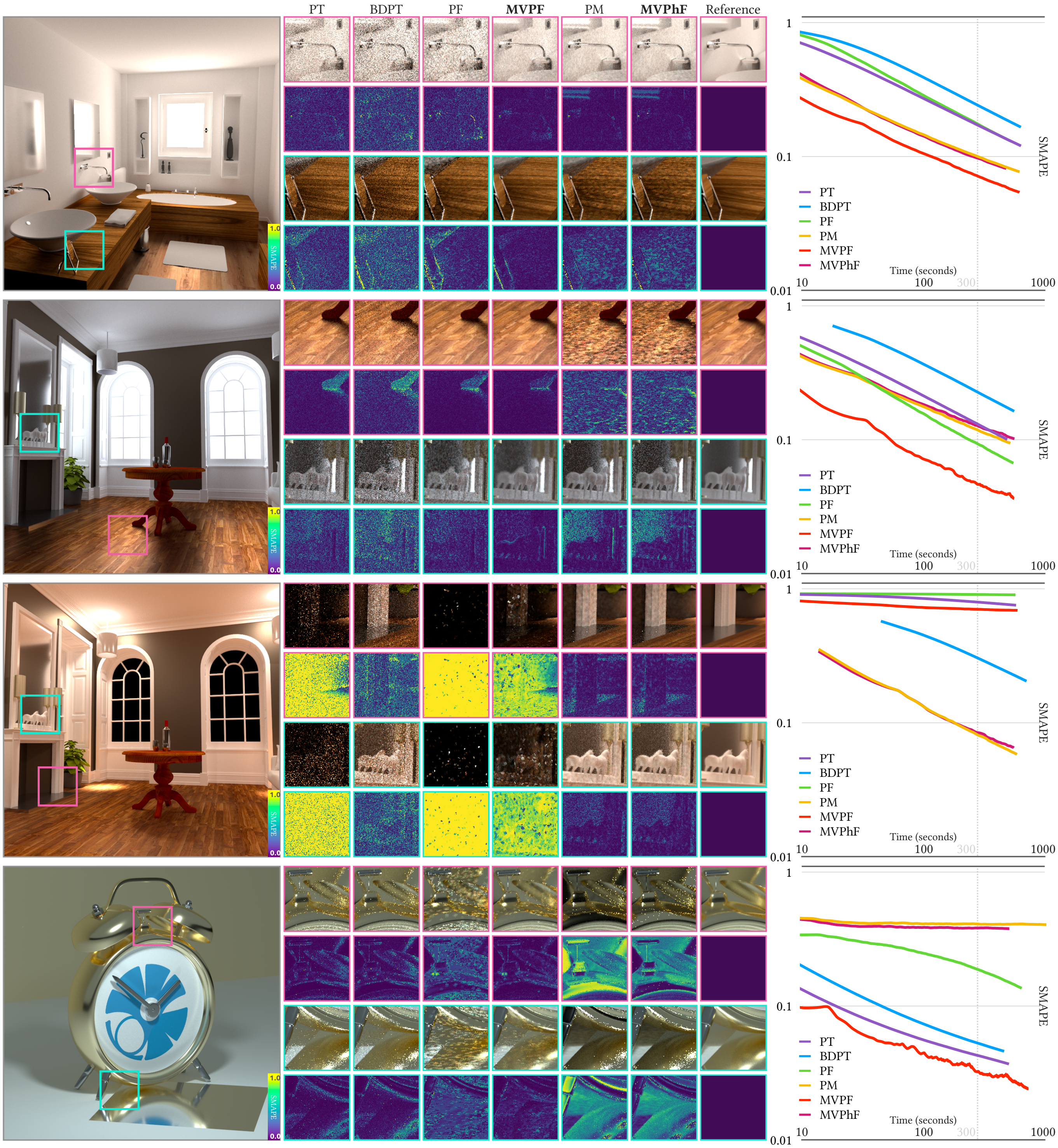


Fig. 1. An equal-time comparison of path tracing (PT), bidirectional path tracing (BDPT), single vertex path filtering (PF), photon mapping (PM), multi-vertex path filtering (MVPF), and multi-vertex photon filtering (MVPhF) on four additional scenes that cover various real-world lighting scenarios.