Supplemental: Segment-based Light Transport Simulation

WENYOU WANG, University of Waterloo, Canada
REX WEST, Aoyama Gakuin University, Japan
TOSHIYA HACHISUKA, University of Waterloo, Canada

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

Additional Key Words and Phrases: rendering, physically-based rendering, path sampling, multiple importance sampling

ACM Reference Format:
Wenyou Wang, Rex West, and Toshiya Hachisuka. 2025. Supplemental: Segment-based Light Transport Simulation. *ACM Trans. Graph.* 44, 4 (August 2025), 5 pages. https://doi.org/10.1145/3730847

S1 SEGMENT PATH INTEGRAL DEFINITIONS

We reformulate the extended path integral [Hachisuka et al. 2012] by changing the original extended path integral into an integral over the segment path space. In the following, we provide the necessary definitions for our segment path integral.

Path. Let $\mathrm{x}_i, \mathrm{y}_i \in \mathcal{M}$. An extended path $\bar{x}$ of length k is defined as:

$$\bar{x} = \mathrm{x}_1\mathrm{y}_1\mathrm{x}_2\mathrm{y}_2 \cdots \mathrm{x}_{k-1}\mathrm{y}_{k-1}\mathrm{x}_k\mathrm{y}_k, \tag{S1}$$

where $1 < k < \infty$. By pairing the vertices x_i and y_i together as a segment s_i, we can rewrite the extended path of vertices as a path of segments:

$$\begin{aligned}\bar{x} =& \mathrm{x}_1\mathrm{y}_1\mathrm{x}_2\mathrm{y}_2 \cdots \mathrm{x}_k\mathrm{y}_k, &\text{(S2)}\\ =& \mathrm{s}_1\mathrm{s}_2 \cdots \mathrm{s}_k, &\text{(S3)}\end{aligned}$$

where each segment $\mathrm{s}_i = (\mathrm{x}_i, \mathrm{y}_i)$ represents a pair of consecutive vertices.

Space. We define $\mathcal{S}_k$ as the space of segment paths of length k. The total segment path space $\mathcal{S}$ is then:

$$\mathcal{S} = \bigcup_{k=1}^{\infty} \mathcal{S}_k. \tag{S4}$$

Note that although the space is now redefined in terms of segments, the underlying space remains the same as in the extended path integral.

Measure. While we now integrate over a sequence of segments rather than vertices—emphasizing the local connectivity between consecutive vertices—the product measure μ_k^* for a segment path of length k is equivalent to the product measure μ_k' for an extended path of length k:

$$\mu_k^* = \prod_{i=1}^{k} d(\mathrm{s}_i) = \prod_{i=1}^{k} \left(d(\mathrm{x}_i) \cdot d(\mathrm{y}_i)\right). \tag{S5}$$

Measurement Contribution Function. We rewrite the measurement contribution of the extended path integral by replacing the vertex functions with segment functions, as shown in Equation 4. However, as discussed in Section 3.1, the kernel and throughput functions in the original extended path integral cannot be directly expressed in terms of segments. Therefore, we extend the definitions of these functions to handle segments.

S2 BRIDGE SEGMENT SAMPLERS

Given segment sequences starting from the camera and light sources, a natural extension is to introduce a segment equivalent of connections as in vertex-based samplers [Veach and Guibas 1994] to combine different segment sequences. We refer to these segments as bridge segments to distinguish them from the classical definition of deterministic connections.

A simple strategy to sample a bridge segment s_i is to sample two vertices in the support of the kernel function around the endpoints $\mathrm{z}_{i-1} \in (\mathrm{x}_{i-1}, \mathrm{y}_{i-1})$ and $\mathrm{z}_{i+1} \in (\mathrm{x}_{i+1}, \mathrm{y}_{i+1})$ of two other segments s_{i-1} and s_{i+1} that we want to connect,

$$p(\mathrm{s}_i|\mathrm{s}_{i-1}, \mathrm{s}_{i+1}) = p(s|\mathrm{z}_{i-1}, \mathrm{z}_{i+1}) = p_K(\mathrm{x}_i|\mathrm{z}_{i-1})p_K(\mathrm{y}_i|\mathrm{z}_{i+1}), \tag{S6}$$

where the perturbation PDF $p_K(\mathrm{x}|\mathrm{z}) = |\mathcal{K}(\mathrm{z})|^{-1}$ for a simple, uniform kernel over the region $\mathcal{K}(\mathrm{z})$ centered at z. This strategy importance samples the two kernel terms at z_{i-1} and z_{i+1}, but not the geometry term or throughput terms, similarly to connections in a vertex-based formulation.

In the spirit of next event estimation in path tracing and light tracing, we can sample a bridge segment to connect an existing segment s_{i-1} to the camera,

$$p(\mathrm{s}_i|\mathrm{s}_{i-1}) = p(\mathrm{s}|\mathrm{z}_{i-1}) = p_W(\mathrm{x}_i)p_K(\mathrm{y}_i|\mathrm{z}_{i-1}), \tag{S7}$$

where $p_W(\mathrm{x}_i)$ is the PDF for sampling a vertex x_i on the sensor, and to light sources,

$$p(\mathrm{s}_i|\mathrm{s}_{i-1}) = p(\mathrm{s}|\mathrm{z}_{i-1}) = p_L(\mathrm{x}_i)p_K(\mathrm{y}_i|\mathrm{z}_{i-1}), \tag{S8}$$

where $p_L(\mathrm{x})$ is the PDF for sampling a vertex x_i on light sources.

S3 MARGINAL MULTIPLE IMPORTANCE SAMPLING

The MMIS estimator introduced by West et al. [2022] can be derived by introducing a set of T *technique spaces* $\mathcal{T}_i$ and a weighting

Authors' addresses: Wenyou Wang, University of Waterloo, Canada, wenyouwang@outlook.com; Rex West, Aoyama Gakuin University, Japan, rexwest@gmail.com; Toshiya Hachisuka, University of Waterloo, Canada, toshiya.hachisuka@uwaterloo.ca.

function $w(t,x)$ satisfying $\sum_{i=1}^{T}\int_{\mathcal{T}_i} w(t,x) = 1$ for every x, and partitioning the integral I into a sum:

$$I = \int_{\mathcal{X}} \underbrace{\sum_{i=1}^{T}\int_{\mathcal{T}_i} w(t,x)\,\mathrm{d}t}_{=1} f(x)\,\mathrm{d}x = \sum_{i=1}^{T}\int_{\mathcal{T}_i}\int_{\mathcal{X}} w(t,x) f(x)\,\mathrm{d}x\,\mathrm{d}t\,. \tag{S9}$$

A multi-sample MIS estimator of Eq. (S9) is then simply,

$$\langle I\rangle_{\mathrm{MMIS}} = \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{w(t_{i,j},x_{i,j}) f(x_{i,j})}{n_i p_i(t_{i,j},x_{i,j})}\,, \tag{S10}$$

where we select $n_i > 0$ technique-sample pairs $(t_{i,j},x_{i,j})$ from the i^{th} technique-space $\mathcal{T}_i$, and the balance heuristic is,

$$w(t_{i,j},x_{i,j}) = \frac{n_i p_i(t_{i,j},x_{i,j})}{\sum_{i'=1}^{T} n_{i'}\int_{\mathcal{T}_{i'}} p_{i'}(t',x_{i,j})\,\mathrm{d}t'}\,. \tag{S11}$$

The integral in the denominator of the balance heuristic weight function (S11) is computationally intractable, so West et al. [2022] propose to stochastically approximate it using Monte Carlo integration,

$$\langle I\rangle^{BH}_{\mathrm{MMIS}} = \sum_{i=1}^{T}\sum_{j=1}^{n_i} \underbrace{\frac{n_i p_i(t_{i,j},x_{i,j})}{\sum_{i'=1}^{T} n_{i'}\int_{\mathcal{T}_{i'}} p_{i'}(t',x_{i,j})\,\mathrm{d}t'}}_{=w(t_{i,j},x_{i,j})} \frac{f(x_{i,j})}{n_i p_i(t_{i,j},x_{i,j})} \tag{S12}$$

$$\approx \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{n_i p_i(t_{i,j},x_{i,j})}{\sum_{i'=1}^{T} n_{i'}\frac{1}{n_{i'}}\sum_{j'=1}^{n_{i'}}\frac{p_{i'}(t_{i',j'},x_{i,j})}{p_{i'}(t_{i',j'})}} \frac{f(x_{i,j})}{n_i p_i(t_{i,j},x_{i,j})} \tag{S13}$$

$$= \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{f(x_{i,j})}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(x_{i,j}|t_{i',j'})}\,, \tag{S14}$$

which is an approximation of the balance heuristic, but still an unbiased estimator for I.

S4 DERIVATIONS FOR MMIS ESTIMATORS

For some rendering scenarios we may have more than one sampling technique, and some of these sampling techniques may be conditioned on auxiliary random variables that were stochastically sampled. West et al. [2022] introduce marginal path sampling (MPS), an general-purpose estimation framework to handle such situations for *vertex-based* path sampling. We can derive a similar general purpose estimator for the segment path integral (3) by extending the domain of integration over $\bar{T}$ technique-spaces $\bar{\mathcal{T}}$ comprised of auxiliary variable sequences $\bar{\mathrm{t}}$,

$$J_{\rho k} = \int_{\mathcal{S}_k} \underbrace{\sum_{i=1}^{\bar{T}}\int_{\bar{\mathcal{T}}_i} w(\bar{\mathrm{t}},\bar{\mathrm{s}})\,\mathrm{d}\bar{\mathrm{t}}}_{=1} g_\rho(\bar{\mathrm{s}})\,\mathrm{d}\bar{\mathrm{s}} = \sum_{i=1}^{\bar{T}}\int_{\bar{\mathcal{T}}_i}\int_{\mathcal{S}_k} w(\bar{\mathrm{t}},\bar{\mathrm{s}}) g_\rho(\bar{\mathrm{s}})\,\mathrm{d}\bar{\mathrm{s}}\,\mathrm{d}\bar{\mathrm{t}}\,, \tag{S15}$$

for which we can construct an MMIS estimator,

$$\langle J_{\rho k}\rangle = \sum_{i=1}^{\bar{T}}\sum_{j=1}^{n_i} \frac{w(\bar{\mathrm{t}}_{i,j},\bar{\mathrm{s}}_{i,j}) g_\rho(\bar{\mathrm{s}}_{i,j})}{n_i p_i(\bar{\mathrm{t}}_{i,j},\bar{\mathrm{s}}_{i,j})}\,, \tag{S16}$$

and use the approximate balance heuristic weighting function,

$$\langle J_{\rho k}\rangle_{\mathrm{BH}} = \sum_{i=1}^{\bar{T}}\sum_{j=1}^{n_i} \frac{g_\rho(\bar{\mathrm{s}}_{i,j})}{\sum_{i'=1}^{\bar{T}}\sum_{j'=1}^{n_{i'}} p_{i'}(\bar{\mathrm{s}}_{i,j}|\bar{\mathrm{t}}_{i',j'})}\,. \tag{S17}$$

The estimator in Eq. (S17) offers a flexible framework for implementing various segment-based light transport algorithms.

Recursive form. Similarly, we can derive a general purpose MMIS estimator for the recursive form (7) of the segment path integral (3) by extending the domain of integration over $\bar{T}$ technique-spaces $\bar{\mathcal{T}}$ comprised of auxiliary variable sequences $\bar{\mathrm{t}}$,

$$L(\mathrm{s}) = L_e(\mathrm{s}) + \int_{\mathcal{S}_1} \underbrace{\sum_{i=1}^{\bar{T}}\int_{\bar{\mathcal{T}}_i} w(\bar{\mathrm{t}},\bar{\mathrm{s}})\,\mathrm{d}\bar{\mathrm{t}}}_{=1} K(\mathrm{s},\mathrm{s}') f_r(\mathrm{s},\mathrm{s}') G(\mathrm{s}') L(\mathrm{s}')\,\mathrm{d}\mathrm{s}'\,, \tag{S18}$$

for which we can construct an MMIS estimator,

$$\langle L(\mathrm{s})\rangle = L_e(\mathrm{s}) + \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{w(\mathrm{t}_{i,j},\mathrm{s}'_{i,j}) K(\mathrm{s},\mathrm{s}'_{i,j}) f_r(\mathrm{s},\mathrm{s}'_{i,j}) G(\mathrm{s}'_{i,j}) \langle L(\mathrm{s}'_{i,j})\rangle}{n_i p_i(\mathrm{t}_{i,j},\mathrm{s}'_{i,j})}\,, \tag{S19}$$

and apply the approximate balance heuristic weighting function,

$$\langle L(\mathrm{s})\rangle_{\mathrm{BH}} = L_e(\mathrm{s}) + \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{K(\mathrm{s},\mathrm{s}'_{i,j}) f_r(\mathrm{s},\mathrm{s}'_{i,j}) G(\mathrm{s}'_{i,j}) \langle L(\mathrm{s}'_{i,j})\rangle_{\mathrm{BH}}}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(\mathrm{s}'_{i,j}|\mathrm{t}_{i',j'})}\,, \tag{S20}$$

Similarly, we can construct an MMIS estimator for the pixel-forming equation (8),

$$\langle J_\rho\rangle = \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{w(\mathrm{t}_{i,j},\mathrm{s}_{i,j}) W_\rho(\mathrm{s}_{i,j}) G(\mathrm{s}_{i,j}) \langle L(\mathrm{s}_{i,j})\rangle}{n_i p_i(\mathrm{t}_{i,j},\mathrm{s}_{i,j})}\,, \tag{S21}$$

and with approximate balance heuristic applied,

$$\langle J_\rho\rangle_{\mathrm{BH}} = \sum_{i=1}^{T}\sum_{j=1}^{n_i} \frac{W_\rho(\mathrm{s}_{i,j}) G(\mathrm{s}_{i,j}) \langle L(\mathrm{s}_{i,j})\rangle_{\mathrm{BH}}}{\sum_{i'=1}^{T}\sum_{j'=1}^{n_{i'}} p_{i'}(\mathrm{s}_{i,j}|\mathrm{t}_{i',j'})}\,. \tag{S22}$$

Discussion. It is worth noting that, while the underlying formulations for the segment path integral (3) and the recursive-form (8) are equivalent, their respective MMIS estimators are not. The weight of a complete path under the recursive form (S19) is the product of the recursion-local MMIS terms — capturing only a subset of weighting functions that satisfy the extended segment path integral (S15). While recursion-local MMIS weights might not be as optimal as complete path MMIS weights, the computation can be significantly more tractable, particularly for path reuse methods, as we demonstrate in the experimental results.

S5 IMPLEMENTATION DETAILS

Our rendering system includes three key components: segment samplers, clustering algorithms, and propagation. Each iteration of our method involves the following steps: 1) sampling segments using segment samplers, 2) clustering the end vertices of segments, and 3) propagating segment throughput on a per-cluster basis. While our segment-based formulation redefines some parts of the traditional light transport simulation, most components of existing rendering systems (e.g., BSDFs and path samplers) remain compatible, requiring only light abstraction on top of a standard vertex-based renderer. The implementation details of each step are elaborated in

the following subsections, and a high-level pseudocode for prototyping our algorithm is provided in Algorithm 1.

Segment Sampling. We implemented both the sequential segment samplers and the bridge segment sampler, following the sampling processes outlined in Sections S2 and 4.1. The sequential segment samplers are based on traditional path sampling but incorporate the concept of virtual perturbation introduced by Hachisuka et al. [2012]. Unlike traditional path samplers, our approach outputs data as independent, disconnected segments. For the bridge segment sampler, segment pairs are constructed to sample both end vertices. Specifically, for each camera segment, a corresponding light segment is uniformly sampled to form a pair. While this straightforward approach is sufficient for preliminary results, alternative sampling schemes, such as power-based sampling, could also be explored to enhance performance.

Clustering. Before detailing the clustering process, we define the kernel function. Our kernel function is a uniform kernel, with its support determined by the position-normal voxel encoding method, commonly used in spatial hashing. The encoded values of the end vertices for all segments are stored, and clustering is achieved by sorting the vertices based on their encoded keys. This process groups vertices within the same kernel function support. To mitigate clustering artifacts, we apply jittering to the position encoding, similar to Binder et al. [2018].

While this voxel encoding method efficiently facilitates clustering, explicitly evaluating the kernel function presents challenges. To approximate its value, we compute the average position and normal of all vertices within a voxel and construct an imaginary plane based on these averages. The kernel value is then approximated as the reciprocal of the intersection area between this plane and the voxel. This approximation may introduce a darkening artifact, akin to the overestimated kernel size issue in photon density estimation. However, this bias can be mitigated by progressively reducing the voxel size.

Propagation. By replacing the range query in filtering with clustering results, we can effectively produce a graph where clusters represent nodes and edges correspond to segments. This graph structure enables an iterative estimation of the recursive MMIS estimator. Below, we present algorithmic details and an intuitive discussion of the propagation algorithm.

Ideally, before propagation, it is essential to pre-compute the MMIS weights for each segment to avoid redundant computations, as the weighting function remains unchanged during propagation. This weighting function requires consideration of all potential techniques, as outlined in Algorithm 2. Since each cluster is associated with a fixed set of segment sampling techniques, and each segment could only be sampled by techniques from two clusters due to the dual kernels Fig. 3, the estimation of the MMIS weight function for each segment involves only the two relevant clusters. This greatly simplifies the implementation and complexity.

Once the MMIS weights are computed, the system is ready for propagation. For each cluster, and for each outgoing segment within the cluster, the estimator in Eq. (S20) can be used to update the outgoing radiance estimate. This updated estimate is then propagated along the segment as an incoming radiance estimate for another cluster. Iteratively performing this process propagates the updated estimates throughout the entire graph. The propagated estimates are subsequently gathered at the camera using the estimator in Eq. (S22). For further details on MMIS weight computation and propagation, refer to Algorithm 2 and Algorithm 3.

Discussion. This propagation process effectively forms a set of paths and computes their contribution at the same time. A key advantage of the propagation process for estimating the recursive MMIS estimator is its ability to share intermediate iterative results, thereby avoiding redundant computations. Radiance estimates of sub-graphs are aggregated at each cluster (node) once and reused, eliminating repetitive work for shared sub-graphs although the overall algorithm still has a worst-case complexity of $O(n^2)$ in the number of segments.

Algorithm 1: Our Method

Input: A set of segment sampling techniques *samplers*
Input: Number of segment (path) samples for each sampler n_i
Input: Number of propagation iterations k

```
segments ← ∅
/* Segment Sample Generation                                   */
for i ← 1 to n do
    s ← sampler[i].sample(n_i)                  // draw n_i samples
    segments.push(s)
clusters = clustering(segments) /* Build Clusters              */
/* MMIS Weight Comp                                            */
for cluster ∈ clusters do
    for segment ∈ cluster.segments do
        segment.w ← MMISWeight(segment, cluster)
/* Propagation                                                 */
for segment ∈ segments do
    segment.in ← L(segment)
for 1 to k do
    for cluster ∈ clusters do
        for segment ∈ cluster.segments do
            segment.out ← propagation(segment, cluster)
    for segment ∈ segments do
        segment.in ← segment.out
/* Final Gathering                                             */
for pixel ∈ image do
    segments ← getSegmentsForPixel(pixel, clusters)
    for segment ∈ segments do
        image[pixel].accumulate(segment.in)
```

S6 ADDITIONAL RESULTS FOR CONVERGENCE AND PERFORMANCE

In addition to the convergence results with progressive kernel size reduction shown in Fig. 4, we also present the convergence plots for our method with the filtering kernel size fixed at its initial value in Fig. S2. In this configuration, our method demonstrates the fastest

Algorithm 2: Pre-compute MMIS Weight

Function MMISWeight(s, $cluster$):
  $p_{sum} \leftarrow 0$
  for $T \in cluster.T$ **do**
    for $t \in T$ **do**
      $p \leftarrow p_c(s,t)$
      $p_{sum} \leftarrow p_{sum} + p$
  return $\frac{1}{p_{sum}}$

Algorithm 3: Propagation

Function Propagation(s, $cluster$):
  $out \leftarrow 0$
  for $s' \in cluster.segments$ **do**
    $out \leftarrow out + s'.w \cdot K(s,s') f_r(s,s') G(s') s'.in$
  $s.out \leftarrow s.out + out$
  return $s.out$

	Bathroom	*Saloon*	*Kitchen*	*DiscoBox*
				
Segment Generation	0.53 sec	0.50 sec	0.36 sec	0.41 sec
Clustering	0.55 sec	0.53 sec	0.59 sec	0.96 sec
Area Estimation	0.06 sec	0.04 sec	0.06 sec	0.08 sec
MMIS Weight Comp	1.24 sec	0.69 sec	0.63 sec	1.51 sec
Filtering	5.52 sec	2.23 sec	2.48 sec	7.57 sec
Iteration Time	**7.90 sec**	**3.98 sec**	**4.11 sec**	**10.54 sec**
Segment Storage	1.46 GB	1.46 GB	1.46 GB	2.19 GB
Clustering Storage	865 MB	865 MB	865 MB	1.38 GB

Table 1. Performance breakdown of one iteration of our method for a fixed kernel size, set to the initial size of our progressive variant. We note here that the cost of segment sampling is significantly eclipsed by the overhead of filtering. This leaves room to explore more advanced segment sampling techniques (that may have an increased cost) without adversely affecting per-iteration overhead.

convergence during the initial stage. However, as the convergence process progresses, the bias introduced by the fixed kernel size causes our method to eventually be outperformed by multi-vertex filtering in several scenes.

Detailed breakdowns of computation times and storage cost for our methods with fixed kernel sizes are provided in Fig. S2. We also include an interesting dynamic time breakdown for the progressive kernel reduction process for the warehouse scene in Fig. S1.

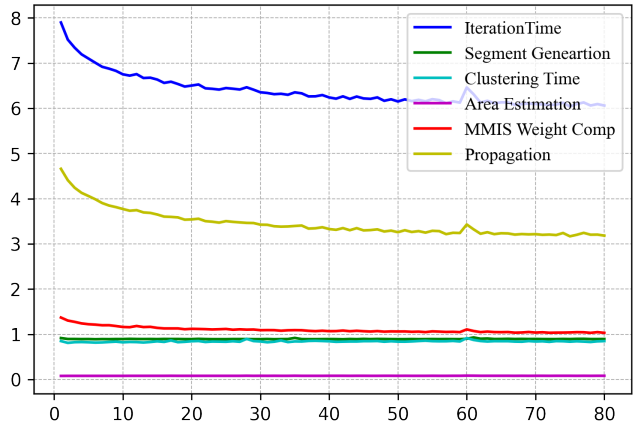

Fig. S1. Time breakdown of each stage in our pipeline for the Warehouse scene with progressive kernel size reduction (x-axis represents samples per pixel (SPP), and the y-axis represents time). During the progressive process, the time required for clustering-based computation remains constant, while the time cost for stages dependent on cluster size (MMIS Weight Computation and Propagation) decreases but does not reach zero, as these operations are still applied to each segment.

REFERENCES

Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2018. Fast Path Space Filtering by Jittered Spatial Hashing. In *ACM SIGGRAPH 2018 Talks* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. ACM, New York, NY, USA, Article 71, 2 pages. https://doi.org/10.1145/3214745.3214806

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A Path Space Extension for Robust Light Transport Simulation. 31, 6 (Jan. 2012), 191:1–191:10. https://doi.org/10/gbb6n3

Eric Veach and Leonidas J. Guibas. 1994. Bidirectional Estimators for Light Transport, Georgios Sakas, Peter Shirley, and Stefan Müller (Eds.). 145–167. https://doi.org/10/gfznbh

Rex West, Iliyan Georgiev, and Toshiya Hachisuka. 2022. Marginal Multiple Importance Sampling. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 42, 8 pages. https://doi.org/10.1145/3550469.3555388

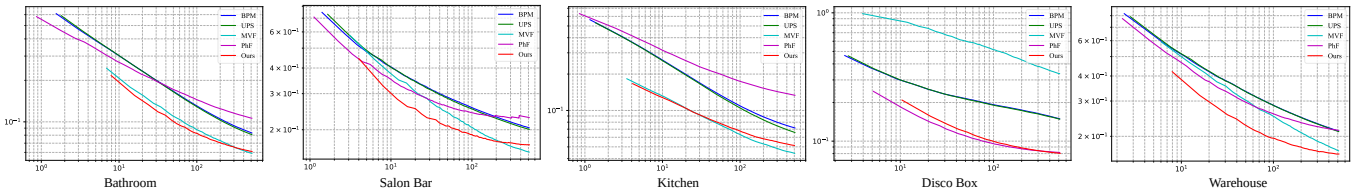

Fig. S2. Equal-time comparison (500s) log-log convergence plots for five test scenes (MAPE vs. Time). Our method performs best during the initial 100 seconds of rendering. However, as rendering progresses, the bias introduced by the fixed kernel size causes the convergence process to slow down, letting MVF catch up in terms of error reduction.