Learning to Cluster for Rendering with Many Lights

YU-CHEN WANG, National Taiwan University, Taiwan
YU-TING WU, National Taiwan University, Taiwan
TZU-MAO LI, MIT CSAIL & University of California San Diego, United States
YUNG-YU CHUANG, National Taiwan University, Taiwan

We present an unbiased online Monte Carlo method for rendering with many lights. Our method adapts both the hierarchical light clustering and the sampling distribution to our collected samples. Designing such a method requires us to make clustering decisions under noisy observation, and making sure that the sampling distribution adapts to our target. Our method is based on two key ideas: a coarse-to-fine clustering scheme that can find good clustering configurations even with noisy samples, and a discrete stochastic successive approximation method that starts from a prior distribution and provably converges to a target distribution. We compare to other state-of-the-art light sampling methods, and show better results both numerically and visually.


Additional Key Words and Phrases: Direct illumination, ray tracing, many-light rendering, optimization theory, reinforcement learning

ACM Reference Format:

1 INTRODUCTION

Rendering with a large number of light sources brings up challenges for importance sampling, since the sampling needs to take the geometry, visibility, light intensity, and material properties into consideration. Two strategies are often applied to reduce the variance: first, to reduce the number of sampling targets to a manageable subset, a clustering step is often applied using a light hierarchy (Fig. 2). Important lights are represented by smaller clusters, and less important ones are approximated by large clusters. Second, existing methods often employ an online learning process that adapts the sampling distribution using collected data. Unfortunately, when the clustering or the sampling distribution does not faithfully represent the importance of lights, existing methods suffer from high variance. In this paper, we present a data-driven solution that can progressively improve both light clustering and sampling using information collected during rendering. Our method is unbiased, provably converging to the target distribution, and supports both direct illumination and virtual point lights.

Fig. 2 shows an example of the importance of clustering. The scene has two groups of triangle lights. For the shading points inside the shell, the lights closer to the shading points are blocked, thus they are only lit by the farther lights. However, most existing methods (e.g., BORAS [Vévodá et al. 2018]) ignore visibility when clustering the lights; therefore they assign fine-grained clustering to the occluded lights, and approximate the important contributors with only one cluster. In contrast, our method learns to cluster using collected data, which allows us to cluster the lights correctly.

Designing an online learning method that simultaneously adapts the sampling distribution and the clustering faces a dilemma: we
need high-quality samples to obtain good clustering, and high-quality clustering to have good samples. We address this challenge with two key ideas: 1) We start from a coarse clustering to accumulate sampling information, and gradually obtain finer clustering as we collect more samples. 2) We use a stochastic approximation method [Robbins and Monro 1951] to update the sampling distribution. This allows us to start from a good prior distribution that leverages the geometry and material information, and provably converges to the target distribution where more factors are involved. We are heavily inspired by the recent work from Pantaleoni [2019] (RLL), which also adapts the cluster while rendering. However, we show that Pantaleoni’s method can often find suboptimal clustering configurations, and does not converge to the target distribution, which leads to visual artifacts (e.g., Fig. 1, top row). We also discuss the connection to reinforcement learning [Dahm and Keller 2017]. We compare to state-of-the-art methods, including Pantaleoni [2019] and Kulla 2018; Liu et al. 2019; Moreau et al. 2019; Pantaleoni 2019; Vévoda et al. 2018].

Many methods learn the importance of light sources in a data-driven way. Samples are collected either in a preprocessing step [Georgiev et al. 2012; Wu and Chuang 2013], or with online updates [Donikian et al. 2006; Fernandez et al. 2002; Pantaleoni 2019; Vévoda et al. 2018]. We build on these approaches and introduce two new ideas: a coarse-to-fine clustering scheme that allows us to make clustering decisions on noisy samples, and a discrete stochastic approximation method that allows our sampling distribution to start from a prior distribution and converge to our target.

**Resampled importance sampling.** Importance resampling methods [Rubin 1987; Talbot et al. 2005] enables sampling of complex distributions without a hierarchy. It first draws samples from a simpler distribution, then samples a subset from the first set by weighting them according to the complex distribution. Resampling has been used for real-time many-lights sampling along with reusing the spatiotemporal neighbors’ sampling distribution [Bitterli et al. 2020]. The hierarchy-free sampling makes these methods more suitable for dynamic lights and more GPU friendly, but these methods do not adapt sampling distribution to visibility in a progressive manner.

**Path guiding.** Some methods build an importance sampling distribution of the hemispherical incoming radiance using collected samples [Dahm and Keller 2017; Lafortune and Willems 1995; Müller et al. 2017, 2019; Pantaleoni 2020; Vorba et al. 2014]. Müller et al.’s adaptive quadtree [2017] is related to our adaptive clustering. These
Hierarchical rendering methods. Hierarchical clustering is often employed in rendering algorithms [Hanrahan et al. 1991; Keller 2001; Overbeck et al. 2009] and they share similarities to our progressive hierarchical refinement. We apply this class of methods to many-light sampling.


3 BACKGROUND: RENDERING WITH DIRECT ILLUMINATION

Given a shading point position \( x \) with a viewing direction \( \omega_o \) and a set of lights \( L \), we are interested in estimating the sum over all the light contributions \( F \):

\[
L(x, \omega_o) = \sum_{l \in L} F(x, \omega_o, l).
\]  

The actual content of the contribution \( F \) depends on the type of light \( l \). If \( l \) is a point light, then \( F \) is the product of the geometry term, visibility, material (i.e., the Bidirectional Scattering Distribution Function), and the light intensity. On the other hand, if \( l \) is an area light, then \( F \) is an integral over the points on the area on the light. \( l \) can also be a virtual point light, generated by light tracing [Keller 1997] or bidirectional path tracing [Davidović et al. 2010; Segovia et al. 2006]. The contribution of a virtual point light corresponds to a point light with a potentially directionally-varying intensity. It is also possible to include the multiple importance sampling weights in the contribution \( F \) [Popov et al. 2015; Walter et al. 2012].

When the size of the set \( L \) is large (say, thousands or millions), evaluating the whole discrete sum for each shading point is not practical. Therefore we rely on Monte Carlo sampling for estimating Equation (1). However, the variance of the Monte Carlo estimator depends on the importance sampling distribution, and a good importance sampling distribution depends on the contribution \( F \).

To importance sample the discrete sum, we need a way to assign an importance value for each light. Doing so individually for each shading-point-light-pair is too expensive: there will be millions of shading points, and thousands or even millions of lights. Instead, existing works often importance sample the lights by clustering them into more manageable non-overlapping subsets \( C(x) \):

\[
L(x) = \sum_{c \in C(x)} \sum_{l \in c} F(x, l) = \sum_{c \in C(x)} F_c(x),
\]  

where \( F_c(x) = \sum_{l \in c} F(x, l) \) and we omit the directional dependency \( \omega_o \) for brevity, without loss of generality. They then estimate the double summation using Monte Carlo sampling:

\[
L(x) \approx \langle L(x) \rangle = \frac{1}{N} \sum_{i=1}^{N} F(x, l_i),
\]  

where \( p(c_i|\omega_o) \) is the probability of choosing the cluster \( c_i \) given the shading point \( x \), and \( p(l_i|\omega_o, c_i) \) is the probability for choosing the light \( l_i \in c_i \) given the shading point and the cluster (or probability density if \( l_i \) is an area light). Sometimes the sampling can be done deterministically, e.g., by evaluating all clusters [Walter et al. 2005]. Sometimes the probabilities are independent of \( x \), e.g., we can choose \( p(l_i|\omega_o, c_i) \) based on the light intensity alone.

In addition to applying clustering to lights, we can also apply clustering to shading points, by letting a group of shading points share the same or similar light sampling distribution [Donikian et al. 2006; Georgiev et al. 2012; Vévoda et al. 2018; Walter et al. 2006; Wu and Chuang 2013]. To come up with the importance sampling distributions \( p(l_i|\omega_o, c_i) \) and \( p(c_i|\omega_o) \), existing work observed that there are often only a few lights that are important for a shading point. Typically, they use a spatial hierarchy to group the lights based on their spatial proximity. Each node on the spatial hierarchy represents a group of lights. Both clustering and sampling can then be done using an approximated contribution of the nodes in the spatial hierarchy. For example, Walter et al. [2005] cluster the lights by thresholding an error bound of the clustering contribution \( F_e \) – which can be quickly estimated without ray tracing by using the bounding box of the lights of a node. Yuksel [2019] adopts a similar error bound, but instead of using it for clustering, they use it for sampling by probabilistically selecting the children of the tree based on the error bound.

Encoding visibility information in the hierarchical structure for importance sampling is hard [Durand et al. 1997; Fernandez et al. 2002]. Therefore, modern data-driven methods collect the visibility information using Monte Carlo samples [Bitterli et al. 2020; Donikian et al. 2006; Georgiev et al. 2012; Vévoda et al. 2018; Wu and Chuang 2013], either in a preprocessing stage or in an online setting, to adapt the sampling distribution. These methods still do not adapt the clustering configuration \( C(x) \).

Pantaleoni [2019] further adapts the clustering configuration \( C(x) \) during rendering. They maintain estimated importance and a fixed cut size for each cluster, and use a split-collapse algorithm to adjust the cluster configuration by simultaneously splitting a cluster \( c \) with the largest contribution into its children and merging two clusters with the smallest contribution. We found that their method often gets stuck in a suboptimal clustering configuration, and their estimated importance does not converge to the target distribution, leading to artifacts in rendering.

Our work builds on the online learning methods above [Donikian et al. 2006; Pantaleoni 2019; Vévoda et al. 2018]. We share the distribution of light importance among close-by shading points. We adapt both the sampling distribution and clustering over the light hierarchy during rendering. This requires us to address two challenges: 1) clustering under noisy observation and 2) adapting the sampling distribution by taking both the information provided by the hierarchical structure, and the collected samples into consideration, while provably converging to the target distribution.
4 METHOD

Overview. Fig. 3 shows an overview of our algorithm that jointly adapts the sampling distribution and light clustering. Our method is based on two key ideas: 1) To gather reliable statistics, we start from a coarse cut and refine it over the iterations. 2) We apply stochastic approximation to update the statistics for our sampling distribution, making it provably converge to the target distribution. During a preprocessing phase, we subdivide the 3D scene into cells, and for each cell, we maintain a table of the approximated mean and variance over a cut on the light source hierarchy. We importance sample the cluster based on the estimated mean, and sample the lights within the cluster using a standard method [Yuksel 2019]. We then probabilistically expand the clusters based on their variance and how frequently they are visited, and iterate the process. In this way, we provide an unbiased, progressive, and data-driven solution for importance sampling with many lights in a scene. We next detail each step of our algorithm.

4.1 Initialization

Light hierarchy construction and shading point clustering. Given a 3D scene, we first build a light hierarchy using an orientation-aware bounding volume hierarchy [Estevez and Kulla 2018]. To cluster the shading points, we further partition the 3D scene by building another 3D bounding volume hierarchy based on the positions and normals of the shading points [Wu et al. 2015]: we choose a cut on the scene bounding volume hierarchy based on the surface areas of the nodes in the hierarchy.

Light clustering initialization. We construct a global light clustering based on the total power of each light cluster. We start from a coarse light clustering configuration by limiting the cut size to a small number (usually 4 or 8). We initialize the light clustering for each scene partition on demand.

4.2 Learning to Sample and Cluster

During rendering, we use Monte Carlo sampling to sample a light from the hierarchy (Equation (3)). We first sample a cluster according to the probability \( p(c|x) \), then sample a light inside the cluster according to \( p(l|c, x) \). For sampling a light inside the cluster, we rely on the stochastic lightcuts algorithm [Yuksel 2019] (Appendix A). We sample the cluster by maintaining an approximated importance \( Q_t^c \) for each node \( c \), and sample proportionally to the importance:

\[
p(c|x) \propto Q_t^c.
\]

We want our sampling distribution to start from a good initial guess, then converges to the target distribution \( Q_t^c = F_t(x) \), where \( F_t(x) \) is the contribution of the cluster \( c \) evaluated at shading point \( x \). Therefore, we learn the approximated importance using a stochastic approximation algorithm, commonly used in reinforcement learning. We start from an initial guess provided by the stochastic lightcuts method, which incorporates the geometry and the material terms, but ignores the visibility. We then iteratively update the estimates using an exponential moving average:

\[
Q_{t+1}^c = (1 - \alpha_t) Q_t^c + \alpha_t F_t(x),
\]

where \( L_0(c, x) \) is an upper bound estimate of the contribution for node \( c \) (Appendix A), \( \alpha_t \) is a learning rate (step size), and \( F_t(x) \) is the Monte Carlo contribution of sampling cluster \( c \). For each iteration \( t \), we send out \( n_t \) samples, and update the tables after the iteration is done. We also collect the variance statistics from the samples for later use.

Since our update is stochastic (\( F_t(x) \) is a random variable), the learning rate schedule \( \alpha_t \) needs to be chosen carefully for the successive approximation above to converge to the expectation \( F_t(x) \). For example, the constant learning rate \( \alpha_t = \alpha \) adopted by Panatleoni [2019] will not always converge. By contrast, it is known [Robbins and Monro 1951; Sutton and Barto 2018] that the following conditions ensure the update above to converge, i.e., \( \lim_{t \to \infty} Q_t^c = F_t(x) \) with probability 1:

1. \( \langle F_t(x) \rangle \) is an unbiased estimator of \( F_t(x) \).
2. The variance and mean of \( \langle F_t(x) \rangle \) are bounded.
3. \( \sum_{t=1}^{\infty} \alpha_t = \infty \) and \( \sum_{t=1}^{\infty} \alpha_t^2 < \infty \).

To satisfy the criteria, we set the learning rate \( \alpha_t = \frac{1}{\beta t} \), where \( \beta \) and \( \omega \) are parameters that will be given in Section 6.1. We make

---

Fig. 3. Overview of our method. Given a scene and a light hierarchy, our method first spatially partition the scene into cells. Each cell stores a cut on the light hierarchy to represent the clustering, and we initialize them with a coarse cut. Each cluster on the cut maintains the estimated importance of the lights, and a variance estimate. To sample a light, we first choose a cluster with probability proportional to its importance. We then sample a light within the cluster using traditional stochastic lightcuts [Yuksel 2019], and update the statistics using a stochastic approximation algorithm. After a round of sampling, we loop over the clusters and probabilistically expand the nodes that have large variances and are visited often. We then iterate the process.
Learning to Cluster for Rendering with Many Lights

5 RELATION TO REINFORCEMENT LEARNING

Previous work has shown the connections between Monte Carlo rendering and reinforcement learning [Dahm and Keller 2017; Pantaleoni 2019]. Here, we make the connection more explicit by showing that rendering approximates a particular kind of action-value function, and show the relation to our update rule. This means that our method can potentially be applied to other rendering problems, which can be solved by reinforcement learning.

In reinforcement learning, an agent is tasked to perform an action \( a \) on the current state \( s \), from a policy \( \pi(s) \) that maps states to actions. For each action, the agent gets a reward \( r(s, a) \). After an action is done, the agent is transferred to another state \( s' \) with probability \( p(s', s) \). The long-term reward \( Q^\pi(s, a) \) for taking an action under a fixed policy \( \pi \) is defined by the Bellman expectation equation:

\[
Q^\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} Q^\pi(s', a'),
\]

where \( \gamma \) is the discount factor that weights the recursive rewards. Traditionally, the goal of an agent is to maximize the accumulated reward by solving for the best policy \( \pi^* \) (s) that satisfies the Bellman optimal equation \( Q^{\pi^*}(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} Q^{\pi^*}(s', a') \).

To solve the optimal policy, we need to solve for the \( Q \) function.

Dahm and Keller [2017] noticed the structural similarity between the Bellman expectation equation above and a discretized rendering equation. We can treat \( Q \) as the radiance, \( s \) as a point on a surface, \( a \) as a direction, \( r \) as emission and set \( \gamma = 1 \). The state transition from \( s \) to \( s' \) is the deterministic ray tracing, thus the probability of reaching the next position \( s' \) given \( s \) and \( a \) must be 1. We further replace the policy probability \( p_\pi \) with a kernel \( k(a, a', s, s') \) that includes the BRDFs and geometry terms. The equation then becomes a discretized version of the rendering equation:

\[
Q^\pi(s, a) = r(s, a) + \gamma \sum_{a'} k(a, a', s, s') Q^\pi(s', a').
\]

Instead of finding the policy that maximizes the rewards, in rendering, we are interested in finding a policy that is proportional to the target action-value: \( p_\pi(a|s) \propto Q^\pi(s, a) \). Notice that the rendering equation is recursive, so we can solve it using a fixed-point iteration. The following stochastic approximation converges to the target \( Q^\pi \) if the conditions in Section 4.2 are satisfied:

\[
Q(s_t, a_t) \leftarrow (1 - \alpha_t) Q(s_t, a_t) + \gamma \alpha_t \left( r(s_t, a_t) + \frac{k}{p_\pi} Q(s_{t+1}, a_{t+1}) \right),
\]

where \( a_{t+1} \) is randomly sampled from the policy, which is proportional to the current action-value estimate, and \( p_\pi \) is the corresponding probability. We omit the arguments of \( k \) and \( p_\pi \).

Our method can also be seen as a reinforcement learning agent. For us, the action is selecting a light cluster and sampling a light on the current lightcut state, the reward is the Monte Carlo contribution, and the discount factor is 0. Setting \( \gamma = 0 \) in Equation 11 above leads to our update equation (Equation 6).

Table 1. We show the number of lights, size of the average cuts, number of active cells in the shading point clusters, and the consumed memory in our method for each scene.
6 RESULTS

6.1 Experiments Set Up

Compared methods and implementation. We compared with stochastic lightcuts (SLC) [Yuksel 2019], resampled importance sampling (RIS) [Bitterli et al. 2020; Talbot et al. 2005], Bayesian online regression (BORAS) [Vévoda et al. 2018] and its variance-aware variant [Rath et al. 2020] (VA-BORAS), reinforcement lightcuts learning (RLL) [Pantaleoni 2019]. For RIS, we implemented ReSTIR [2020] without the spatiotemporal reusing. We found VA-BORAS usually produces similar or better results than BORAS in our experiments. Therefore, we only show the results of VA-BORAS here. The results of BORAS can be found in the supplement. We implemented all methods, including ours, in the PBRT renderer [Pharr et al. 2016]. All results were generated on an Intel Core i7-9700 CPU using 4 cores, and 32GB RAM.

Test scenes. We evaluated our method on direct illumination sampling with a diverse set of indoor and outdoor scenes. The scenes contain very different lighting conditions, with the number of lights ranging from 8 to 90 K. We also tested our method with two additional scenes for demonstrating that our method can be extended to render virtual point lights (VPL) [Keller 1997] for global illumination. We show four direct illumination comparisons (Fig. 1, Fig. 4, Fig. 6) and one VPL comparison (Fig. 5) in the main paper. Table 1 shows related statistics of these scenes. We include comparisons of other scenes in the supplementary materials.

Hyperparameters. We set the learning rate $\alpha_t$ for iteration $t$ to $1/\beta t^\omega$ where $\beta = 4$ and $\omega = 6/7$. We divide the shading points into 16384, 32768, or 65536 clusters according to their geometry (Section 4.1).
complexity (Table 1). For direct illumination scenes, we set the initial cut size to 4, and the maximum cut size to 64. For VPL scenes, we set the initial cut size to 8, and the maximum cut size to 128. For all scenes, we set the initial sampling budget $n_0 = 4$ and set $\Gamma = 128$. For other methods, we apply the default hyperparameters provided by the authors.

### 6.2 Comparisons

The **Bathroom** scene (Fig. 1) contains a strong window light, four hanging bright light bulbs, and some dim ceiling lights, making more than 4K area lights in total. For the regions where the bright lights are occluded, such as the walls, the light clustering constructed using the error bound of lightcuts is sub-optimal because of not taking visibility into account. Both reinforcement lightcuts learning (RLL) and our method can learn to refine the light clustering based on previous samples. However, RLL produces visual artifacts since its sampling distribution does not converge to the target.

Fig. 4 shows the **Bedroom** scene and the **Classroom** scene. The **Bedroom** scene has difficult visibility because most regions can only receive illumination from a small group of lights. In this case, the quality of light clustering will have a significant impact on the rendering quality. The small bedside lamp highlighted in green demonstrates such a case. Both Bayesian online regression and its variance-aware version can only learn the sampling distribution of a poor light clustering. Reinforcement lightcuts learning learns better lightcuts but not as good as ours. The **Classroom** demonstrates a case of uniform lighting. Both the ceiling lights and window lights can contribute to most regions of the scene. In this case, our method robustly achieves better image quality than all previous methods across the whole image.

The **Kitchen** scene shown in Fig. 5 demonstrates the capability of our method for handling virtual point lights (VPLs). We traced about 70K VPLs. Our method is only used for clustering and sampling VPLs, while direct illumination is rendered with the default light sampling approach implemented in PBRT. As shown in Fig. 5, on the highlighted backlit and glossy surface, previous methods fail to correctly cluster and sample the important VPLs, producing very noisy images and spike artifacts. Our method can better locate, refine and sample important VPL clusters. As a result, our method significantly outperforms all other methods in terms of both visual quality and relative mean squared error.

**Multiple importance sampling before light selection.** In the previous results, we apply BRDF importance sampling after a light is selected, then combine the result using multiple importance sampling (MIS). Alternatively, we can also sample the BRDF before a light is selected. Table 2 shows a quantitative comparison of our method with two MIS strategies: before and after light selection. The better one is highlighted in bold.

### Table 2. The relative mean square error of our method with two MIS strategies: before and after light selection. The better one is highlighted in bold.

<table>
<thead>
<tr>
<th>Scene</th>
<th>before light selection</th>
<th>after light selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bathroom</td>
<td>0.021</td>
<td>0.013</td>
</tr>
<tr>
<td>Bedroom</td>
<td>0.008</td>
<td>0.008</td>
</tr>
<tr>
<td>Classroom</td>
<td>0.012</td>
<td>0.013</td>
</tr>
<tr>
<td>Parking-Lot</td>
<td>0.047</td>
<td>0.079</td>
</tr>
<tr>
<td>Staircase2</td>
<td>0.004</td>
<td>0.003</td>
</tr>
</tbody>
</table>

![Fig. 6.](image)

**Memory Consumption.** The memory consumption of our method in each scene is listed in Table 1. Each light cluster takes 16 bytes to store the variance, second moment (for updating the variance), visit count $n_c$, and the importance $Q^X(c)$. By contrast, a light cluster in BORAS [Vêvoda et al. 2018] takes 40 bytes in our implementation. Thus our method needs less memory for a single light cluster. The overall memory consumption is proportional to the number of shading point clusters and size of lightcuts.

### 6.3 Ablation Studies

**Initial cut size.** A crucial idea of our method is to start from a coarse clustering and refine, so that we collect more reliable information about the clusters. Fig. 7 shows that starting from a coarse light clustering will accelerate variance reduction and result in a relatively small but good light clustering.

**Benefits of cluster refinement and constraints.** We evaluate the benefits of cluster refinement and having constraints that stop the
refinement after reaching certain criteria in Fig. 8. We show that the refinement is indeed helpful, and having the constraints helps to avoid unnecessary clustering while delivering lower error.

Discussion with reinforcement lightcuts learning. Both our method and reinforcement lightcuts learning (RLL) [Pantaleoni 2019] use data to refine cluster and sample lights. Our contribution is the different clustering rule (CR) and update rule (UR). RLL maintains a fixed clustering size, and adjusts the clustering by splitting and merging at the same time, while we adopt a coarse-to-fine strategy. RLL’s update rule does not guarantee convergence, while our stochastic approximation rule converges to the target. Fig. 9 compares the effectiveness of the four combinations of our/UR’s clustering rules and update rules. The combination of our CR and UR leads to the fewest image artifacts and the lowest rMSE. We found that the non-converging RLL update rule often leads to image artifacts. Table 3 shows that this trend continues in other scenes. In scenes with relatively few lights, such as the Living-Room scene (8) and the SiA-Shelf scene (16), RLL produces comparable results to our method. However, when the number of lights increases, our method robustly produces the best results than other methods.

Fig. 8. We compare the evolution of average clustering size, and relative mean absolute error (rMAE) for our method without cluster refinement (W/O CR), with cluster refinement (W/ CR), and with the constrained cluster refinement (W/ C-CR), where we stop splitting when there is no refinement for a while or when we reach a maximum cut size (Section 4.2). We compare them using the Bathroom scene. We start from a light clustering with $|G_0| = 4$. We set the maximum cut size as 32 and $\Gamma = 128$. The cluster refinement indeed leads to lower error, while the constraints avoid unnecessary clustering and also deliver slightly lower error.

Fig. 7. We compare the evolution of average clustering size and relative mean absolute error ($r$MAE) for our method with different initial cut sizes using the Bathroom scene. In the experiment, we constrain the maximum cut size to be 64. With a smaller initial cut size, our method achieves a lower error while avoiding unnecessary clustering.

7 LIMITATIONS AND FUTURE WORK

Sharp lighting edges. Most methods, including ours, use averaged statistics over shading points to reduce computational cost. In the presence of sharp lighting edges, this makes the sampling distribution suboptimal and leads to higher variance, and causes problems to all methods. However, our clustering refinement will often detect the high variance and put more samples on the lighting edges.

GPU implementation. While our algorithm is trivially parallelizable, our current CPU implementation does not optimize for the minimal thread divergence on a SIMD unit. An efficient GPU implementation may require different data structures and design decisions [Bitterli et al. 2020; Lin and Yuksel 2020; Moreau et al. 2019; Pantaleoni 2019]. For real-time rendering of dynamic lights, adapting the hierarchy over time also introduces extra overhead.

Shading point clustering. While we use a data-driven method to learn light clustering, we do not learn the shading point clustering and rely on heuristics. Learning shading point clustering can be crucial for scenes with very complex geometry and materials.

### Table 3: Relative mean square errors of the ablation study for combinations of clustering rule (CR)/update rule (UR). The one with the smallest error is highlighted in bold.

<table>
<thead>
<tr>
<th>Scene</th>
<th>RLL/RLL</th>
<th>RLL/Ours</th>
<th>Ours/RLL</th>
<th>Ours/Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bathroom</td>
<td>0.024</td>
<td>0.021</td>
<td>0.030</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>Bedroom</td>
<td>0.022</td>
<td>0.062</td>
<td>0.009</td>
<td><strong>0.008</strong></td>
</tr>
<tr>
<td>Classroom</td>
<td>0.046</td>
<td>0.034</td>
<td>0.015</td>
<td><strong>0.013</strong></td>
</tr>
<tr>
<td>Kitchen</td>
<td>0.404</td>
<td>0.491</td>
<td>0.091</td>
<td><strong>0.050</strong></td>
</tr>
<tr>
<td>Living-Room</td>
<td>0.006</td>
<td><strong>0.005</strong></td>
<td>0.018</td>
<td><strong>0.005</strong></td>
</tr>
<tr>
<td>Parking-Lot</td>
<td>0.237</td>
<td>0.076</td>
<td>0.232</td>
<td><strong>0.047</strong></td>
</tr>
<tr>
<td>SiA-Shelf</td>
<td>0.042</td>
<td>0.113</td>
<td>0.151</td>
<td><strong>0.038</strong></td>
</tr>
<tr>
<td>Sanmiguel</td>
<td>3.819</td>
<td>2.407</td>
<td>1.099</td>
<td><strong>0.435</strong></td>
</tr>
<tr>
<td>Staircase</td>
<td>0.013</td>
<td>0.009</td>
<td><strong>0.007</strong></td>
<td><strong>0.007</strong></td>
</tr>
<tr>
<td>Staircase2</td>
<td>0.006</td>
<td>0.005</td>
<td>0.004</td>
<td><strong>0.003</strong></td>
</tr>
</tbody>
</table>


8 CONCLUSION

We have presented an unbiased and progressive light sampling method that can adapt both the clustering and the sampling distributions using collected samples. The key ideas are a coarse-to-fine clustering scheme and a stochastic approximation algorithm for updating the sampling distribution, that can provably converge to the target distribution. Our method is robust to both simple and difficult configurations and introduces minimal overhead. By combining with bidirectional path tracing [Popov et al. 2015] and path guiding, our method can potentially be a crucial component inside a fully data-driven importance sampled rendering algorithm.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments, and the creators of the models and textures used in this paper: the BATHROOM, BEDROOM, KITCHEN, STAIRCASE, STAIRCASE2, LIVING-ROOM scenes, the plants and sofa in Fig. 2, and the blue and green cars in Fig. 6, via Benedikt Bitterli’s rendering resources [Bitterli 2016]; the BMW car and sports car in Fig. 6 and the samiguel scene from PBRT [Pharr et al. 2016]; the buildings and street lamps in Fig. 6 via the Open Research Content Archive [Lumberyard 2017]; the fence (Alan_du_os) in Fig. 6 from https://www.turbosquid.com/; the chair (Beniamus3D), desk (abicippa), ground lamp (hrisitopoov), and ceiling light (Evermotion) in Fig. 4 from https://www.cgtrader.com/; the brick, ground, and floor textures in Fig. 6 (Rob Tuytel) from https://polyhaven.com/textures.

This work was supported in part by the Ministry of Science and Technology and AI Technology and All Vista Healthcare, under grants 110-2221-E-002-124-MY3 and 110-2634-F-002-026. We thank to National Center for High-performance Computing (NCHC) for providing computational and storage resources.

REFERENCES


ACM Trans. Graph., Vol. 40, No. 6, Article 277. Publication date: December 2021.
A STOCHASTIC LIGHTCUTS

Our method contains some components from the stochastic lightcuts algorithm [Yuksel 2019]. For completeness we will describe it here. Given a shading point and a node in the light hierarchy, we want to sample a light inside the node. We do it by traversing the light hierarchy, each time randomly picking one of the children until we reach the leaf. Stochastic lightcuts evaluates an estimation of the hierarchy, each time randomly picking one of the children until we reach the leaf. Stochastic lightcuts evaluates an estimation of the probability of sampling. The upper bound \( L_u \) for both of the children of a node to determine the probability of sampling. The upper bound \( L_u \) is estimated by:

\[
L_u(x, c) = \frac{G_u(x, c)M_u(x, c)L_c}{\Lambda(x, c)},
\]

where \( G_u(x, c) \) is the upper bound of the geometry term without the distance squared term, \( M_u(x, c) \) is an upper bound of the materials [Walter et al. 2005] describe how to compute \( G_u(x, c) \) and \( M_u(x, c) \) for Lambertian, Phong, and Ward BRDFs [Walter 2005], and there are ways to bound them for certain shaders [Vélazquez-Armendáriz et al. 2009; Walter et al. 2012], \( L_c \) is the integral of lights inside the cluster, and \( \Lambda(x, c) \) is the attenuation term:

\[
\Lambda(x, c) = \begin{cases} \frac{1}{d_{\text{min}}(x, c)^2} & \text{if } d_{\text{min}}(x, c) > l_c \ \& \ d_{\text{min}}(x, c') > l_{c'} \\ 1 & \text{otherwise} \end{cases}
\]

where \( d_{\text{min}}(x, c) \) is the minimal distance from \( x \) to the bounding box of cluster \( c \), \( c' \) is the sibling of the cluster \( c \) which shares the same parent, and \( l_c \) and \( l_{c'} \) are the length of the diagonal of the bounding boxes of the corresponding clusters. The attenuation term is designed to eliminate the singularity of the error bound when the point \( x \) is inside or very close to a cluster’s bounding box.

B CONNECTION OF OUR UPDATE RULE TO STOCHASTIC APPROXIMATION

Classical stochastic approximation theory [Robbins and Monro 1951] shows that the following update rule converges with probability 1 to the root \( \theta^* \) of a nondecreasing function \( f \) under a zero-mean noise \( \epsilon_n \) with both bounded mean and variance, if \( f'(\theta^*) > 0 \),

\[
\sum_{t=1}^{\infty} \sigma_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \sigma_t^2 < \infty:
\]

\[
\theta_{t+1} = \theta_t - \alpha_t (f(\theta_t) + \epsilon_n).
\]

If we set \( \theta = Q_t^x(c) \) and \( f(\theta) = \theta - F_c(x) \), the equation above becomes our update rule (Equation (6)), and the root of the function \( f \) is \( F_c(x) \). Furthermore, since the statement above applies individually to each element of the table, the distance between the table and the target \( \Sigma_t Q_t^x(c) - F_c(x) \) scales linearly with the dimensionality of the table. It is also known that the expected difference between \( Q_t^x(c) \) and the root \( F_c \) scales linearly with the variance of the noise \( \epsilon_n \) [Bottou et al. 2018].

C INITIALIZATION AFTER SPLITTING A CLUSTER

It can be shown that after \( t \) iterations our importance table is:

\[
Q_t^x(c) = \left( \prod_{i=1}^{t} (1 - \alpha_i) \right) L_u(x, c) + \sum_{j=1}^{t} \alpha_j \left( \prod_{i=j+1}^{t} (1 - \alpha_i) \right) \langle F_c(x) \rangle_j.
\]

Our learning rate is a monotonically decreasing sequence \( \alpha_t = \frac{1}{\tau t} \). We approximate the product as:

\[
\prod_{i=1}^{t} (1 - \alpha_i) \approx (1 - \alpha_t)^t.
\]

For the two children \( c_1 \) and \( c_2 \), we then need to know approximately how many times they have been visited. Since we use stochastic lightcuts for sampling the children, we know that on expectation \( n_{c_1} \) and \( n_{c_2} \) are proportional to their error bound \( L_u(x, c_1) \) and \( L_u(x, c_2) \). Therefore, for children \( c_1 \) we approximate its visited count \( n_{c_1} \approx \frac{L_u(x, c_1)}{L_u(x, c_1) + L_u(x, c_2)} n_x \), where \( n_x \) is the visit count of its parent. The initialized \( Q_0 \) value is then:

\[
Q_0^x(c) \leftarrow (1 - \alpha_t)^{n_{c_1}} L_u(x, c) + (1 - (1 - \alpha_t)^{n_{c_2}}) Q_0^x(c).
\]