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

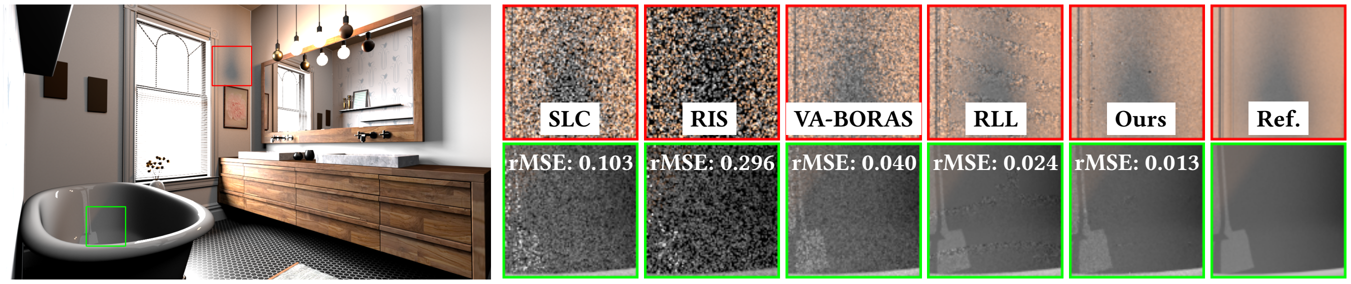


Fig. 1. Bathroom: Equal-time comparison (120s) between stochastic lightcuts (SLC) [Yuksel 2019], resampled importance sampling (RIS) [Bitterli et al. 2020; Talbot et al. 2005], variance-aware Bayesian online regression (VA-BORAS) [Rath et al. 2020], reinforcement lightcuts learning (RLL) [Pantaleoni 2019] and our method. SLC and RIS do not importance sample the actual contribution of light clustering and this causes noise. VA-BORAS's heuristics do not find a good light clustering configuration to learn a distribution on. RLL learns both the clustering and the sampling distributions, but often does not find a good cluster, and their sampling distribution does not converge to the target distribution, leaving artifacts in the shadow regions (top row). Our method learns the clustering using a coarse-to-fine scheme, and our sampling distribution provably converges to the target. Our method achieves the lowest relative mean square error (rMSE) among all compared methods. The reference is rendered by uniform light sampling in 25 hours.

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.

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

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

ACM Reference Format:
Yu-Chen Wang, Yu-Ting Wu, Tzu-Mao Li, and Yung-Yu Chuang. 2021. Learning to Cluster for Rendering with Many Lights. *ACM Trans. Graph.* 40, 6, Article 277 (December 2021), 10 pages. https://doi.org/10.1145/3478513.3480561

Authors' addresses: Yu-Chen Wang, National Taiwan University, No. 1, Section 4, Roosevelt Rd, Taipei, 10617, Taiwan, yucwang@cmlab.csie.ntu.edu.tw; Yu-Ting Wu, National Taiwan University, No. 1, Section 4, Roosevelt Rd, Taipei, 10617, Taiwan, kevincosner@cmlab.csie.ntu.edu.tw; Tzu-Mao Li, MIT CSAIL & University of California San Diego, United States, tzli@ucsd.edu; Yung-Yu Chuang, National Taiwan University, No. 1, Section 4, Roosevelt Rd, Taipei, 10617, Taiwan, cyy@csie.ntu.edu.tw.

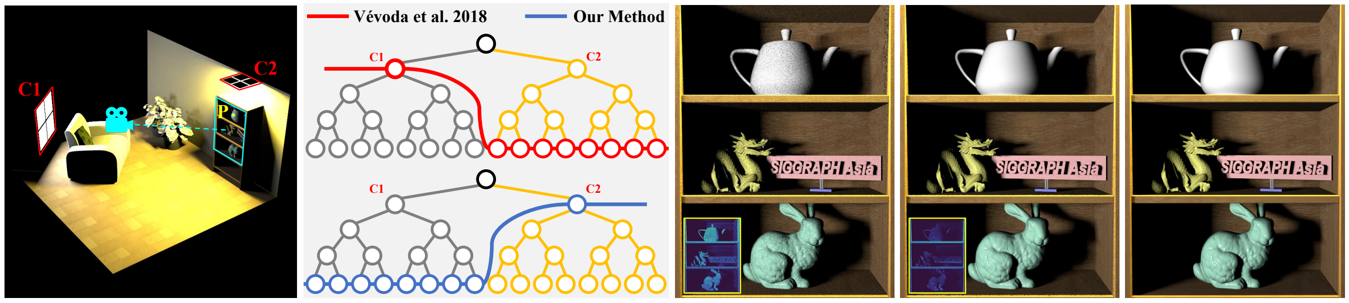
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2021 Copyright held by the owner/author(s).
0730-0301/2021/12-ART277
https://doi.org/10.1145/3478513.3480561

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 shelf, 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 [Vévoda 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



(a) a scene with 16 lights (b) the light clustering configurations at P (c) Vévoda et al. [2018] (d) our method (e) reference

Fig. 2. The importance of light clustering. (a) shows an example scene with 16 area lights, including 8 dull white lights on the left and 8 strong yellow lights above the bookshelf. (b) shows the visualization of the lightcuts at an interest point P. Vévoda et al.'s BORAS method [2018] constructs the cut using heuristics without considering visibility. Thus, it cannot locate the important lights within **C1**. By contrast, our method starts from a cut consisting of **C1** and **C2** and progressively refines the nodes in **C1** according to the information obtained from online learning. (c) and (d) show the rendered results of Vévoda et al.'s method and our method, respectively. The error visualizations are shown at the bottom left corner. (e) shows the reference image. Our method produces an image with much fewer noises than Vévoda et al.'s method because of better adaptive light clustering.

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's algorithm, and show that our method achieves between 1.8× to 7× lower relative mean square error compared to the best methods across different scenes in the same amount of time.

2 RELATED WORK

Rendering with many lights. Earlier work in this category focused on reducing the number of visibility tests [Kok and Jansen 1994; Ward 1994]. Shirley et al. [1996] use an octree to classify lights into important and unimportant ones for each shading point, and focus sampling budget to the important lights. Later methods extend this idea to use a hierarchy to divide lights into multiple clusters (a *cut*). The sampling can be biased [Fernandez et al. 2002; Paquette et al. 1998], unbiased during the hierarchy construction [Walter et al. 2005], or unbiased for each individual shading point [Estevez and Kulla 2018; Liu et al. 2019; Moreau et al. 2019; Pantaleoni 2019; Vévoda et al. 2018; Yuksel 2019].

Some methods cluster lights by sampling the light transport matrix in a preprocessing step [Hašan et al. 2008; Hašan et al. 2007; Huo et al. 2015]. Ou and Pellacini [2011] use a hybrid approach by first grouping the lights using matrix sampling, then forming hierarchical clusters inside each group. These methods do not adapt the clustering once it is formed. Although it might be possible to modify these methods to be progressive, how to do so in an efficient and automatic way is unclear.

Virtual point light methods [Dachsbacher et al. 2014; Keller 1997] convert the indirect illumination problem into direct illumination, by depositing virtual point lights using light tracing. This allows us to treat both direct lighting and global illumination using a unified approach. The visibility can be determined using shadow map [Dachsbacher and Stamminger 2005; Keller 1997; Ritschel et al. 2008] or ray tracing [Kollig and Keller 2006; Popov et al. 2015; Walter et al. 2012]. Our method can be used with virtual point lights and belongs to the ray tracing category.

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 methods focus on the continuous domain, while we address the discrete light sampling problem. Directly adapting path guiding methods to the many-lights problem can be non-trivial, as lights are sparsely distributed spatially. Accounting for the geometry properties, such as orientations and positions of lights, is more difficult in the 5D spatial-directional space.

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-lights sampling.

Reinforcement learning in rendering. Dahm and Keller [2017] observe the similarity between the rendering equation [Kajiya 1986] and the *expected SARSA* reinforcement learning method [Sutton et al. 1998]. Pantaleoni [2019] applies the idea for sampling light clusters. Huo et al. [2020] apply deep reinforcement learning for adaptive sampling and reconstruction. We show the relation of rendering, stochastic approximation [Robbins and Monro 1951], and reinforcement learning in Section 5.

3 BACKGROUND: RENDERING WITH DIRECT ILLUMINATION

Given a shading point position x with a viewing direction ω_o and a set of lights $\mathcal{L}$, we are interested in estimating the sum over all the light contributions F:

$$L(x, \omega_o) = \sum_{l \in \mathcal{L}} F(x, \omega_o, l). \tag{1}$$

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 [Davidovič 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 $\mathcal{L}$ 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 $C(x)$:

$$L(x) = \sum_{c \in C(x)} \sum_{l \in c} F(x, l) = \sum_{c \in C(x)} F_c(x), \tag{2}$$

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

$$L(x) \approx \langle L(x) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{F(x, l_i)}{p(l_i | x, c_i) p(c_i | x)}, \tag{3}$$

where $p(c_i|x)$ is the probability of choosing the cluster c_i given the shading point x, and $p(l_i|x, c_i)$ 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 $p(l_i|x, c_i)$ 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 $p(l_i|x, c_i)$ and $p(c_i|x)$, 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_c – 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.

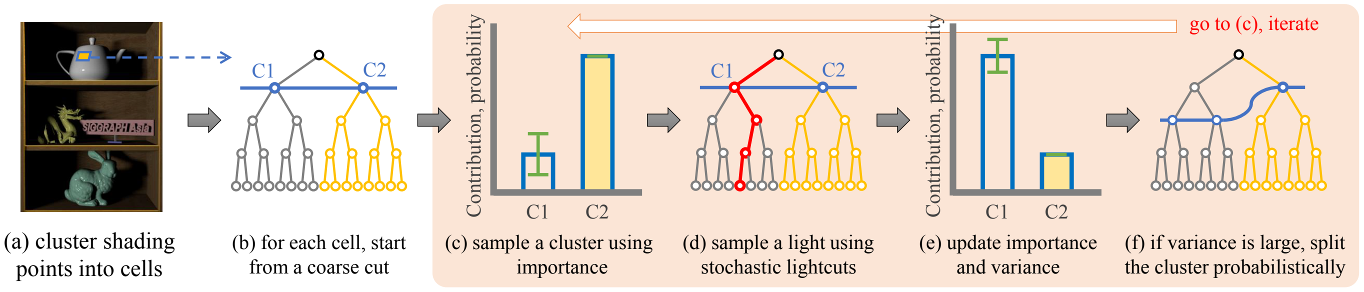

C1
C2
go to (c), iterate
Contribution, probability
(a) cluster shading points into cells
(b) for each cell, start from a coarse cut
(c) sample a cluster using importance
(d) sample a light using stochastic lightcuts
(e) update importance and variance
(f) if variance is large, split the cluster probabilistically

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.

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 5D 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 $Q^x(c)$ for each node c, and sample proportionally to the importance:

$$p(c|x) \propto Q^x(c). \tag{4}$$

We want our sampling distribution to start from a good initial guess, then converges to the target distribution $Q^x(c) = F_c(x)$[1], where $F_c(x)$ 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:

$$Q_0^x(c) = L_u(x,c) \tag{5}$$
$$Q_{t+1}^x(c) = (1-\alpha_t)Q_t^x(c) + \alpha_t\langle F_c(x)\rangle, \tag{6}$$

where $L_u(x,c)$ is an upper bound estimate of the contribution for node c (Appendix A), α_t is a learning rate (step size), and $\langle F_c(x)\rangle$ 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 ($\langle F_c(x)\rangle$ is a random variable), the learning rate schedule α_t needs to be chosen carefully for the successive approximation above to converge to the expectation $F_c(x)$. For example, the constant learning rate $\alpha_t = \alpha$ adopted by Pantaleoni [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., $\lim_{t\to\infty} Q_t^x(c) = F_c(x)$ with probability 1:

1. $\langle F_c(x)\rangle$ is an unbiased estimator of $F_c(x)$.
2. The variance and mean of $\langle F_c(x)\rangle$ 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 $\alpha_t = \frac{1}{\beta t^{\omega}}$, where β and ω are parameters that will be given in Section 6.1. We make

[1]Since we share the distribution over a group of shading points, the target is the average over the group.

the connection to stochastic approximation explicit in Appendix B, while also showing that the error of the estimated importance scales linearly with the size of the table and variance. Inspired by the error rate above, we set the number of samples per-iteration $n_t = \max\left(\frac{|C_t|}{|C_0|}, 2\right)n_0$, where $|C_t|$ is the number of clusters at iteration t, and n_0 is a user-specified parameter.

In principle, the light contribution estimate should be the Monte Carlo estimation $\langle F_c(x)\rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{F(x,l_i)}{p(l_i|x,c_i)p(c_i|x)}$. However, in practice, we found that when sampling clusters closer to the root of the hierarchy, the upper bound estimate L_u can be significantly inaccurate in the presence of the glossy material due to the loose bounding boxes. Therefore, when updating the table, we approximate $p(l_i|x,c_i)$ with the geometric mean between the actual probability and a uniform distribution over all the lights in the cluster.

After each iteration, we loop through each cluster and decide whether we should split it into its two children or not. Our splitting considers four criteria: 1) The cluster should be more likely to be split when the variance is high. 2) We should reduce the probability of splitting when the cut size is already large. 3) We prefer a cluster configuration in which all clusters have similar variance. 4) We should split the clusters which are visited more often. Thus, we set the probability of splitting a cluster c to be:

$$p_t(c) = \frac{1}{1+\frac{|C_t|}{|C_0|}e^{-\mathrm{Var}_t(c)}}\frac{\mathrm{Var}_t(c)}{\sum_{c'\in C(x)}\mathrm{Var}_t(c')}\left(1-\frac{1}{n_c}\right), \tag{7}$$

where $|C_t|$ is the size of the cut at iteration t and n_c is the number of times we sample cluster c. The first term is a sigmoid function that measures the variance relative to the size of the cut. The second term measures the relative variance of the cluster (we add a small number 10^{-6} to the denominator in practice to avoid division by zero). The third term measures the frequency of visits.

To avoid a large number of clusters, we record the latest iteration t' that the light clustering is updated and stop refining the cluster when $(t-t')/|C_t| > \Gamma$, where t is the current iteration and Γ is a user-defined parameters, or when the number of cluster reaches a maximum cut size.

If we choose to split a cluster, we need to initialize the estimated importance of their children c_1 and c_2. We approximate their importance using:

$$Q_t^x(c_1) = AL_u(x,c_1) + (1-A)Q_t^x(c) \tag{8}$$

where $A = (1-\alpha_t)^{\frac{L_u(x,c_1)}{L_u(x,c_1)+L_u(x,c_2)}n_c}$. c_2 is initialized similarly. We explain this approximation in Appendix C. Variance and the visit count n_{c_1} are initialized to 0.

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 $p(s'|s,a)$. The long-term reward $Q^\pi(s,a)$ for taking an action under a fixed policy π is defined by the Bellman expectation equation:

$$Q^\pi(s,a) = r(s,a) + \gamma\sum_{s'}p(s'|s,a)\sum_{a'}p_\pi(a'|s')Q^\pi(s',a'), \tag{9}$$

where γ 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 $Q^{\pi^*}(s,a) = r(s,a)+\gamma\sum_{s'}p(s'|s,a)\,\mathrm{argmax}_{a'}\,Q^{\pi^*}(s',a')$. 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_π 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:

$$Q^*(s,a) = r(s,a) + \gamma\sum_{a'}k(a,a',s,s')Q^*(s',a'). \tag{10}$$

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: $p_{\hat{\pi}}(a|s) \propto Q^*(s,a)$. 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^* if the conditions in Section 4.2 are satisfied:

$$Q(s_t,a_t) \leftarrow (1-\alpha_t)Q(s_t,a_t) + \gamma\alpha_t\left(r(s_t,a_t) + \frac{k}{p_\pi}Q(s_{t+1},a_{t+1})\right), \tag{11}$$

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

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.

Scene	lights	avg. cut	active	mem. (MB)
Bathroom	4776	11.27	6402 / 16384	1.4
Bedroom	8032	14.24	9606 / 32768	2.2
Classroom	1216	37.21	8570 / 32768	6.1
Parking-Lot	90862	20.42	3893 / 65536	1.5
Kitchen (VPL)	71311	35.05	6282 / 32768	4.2

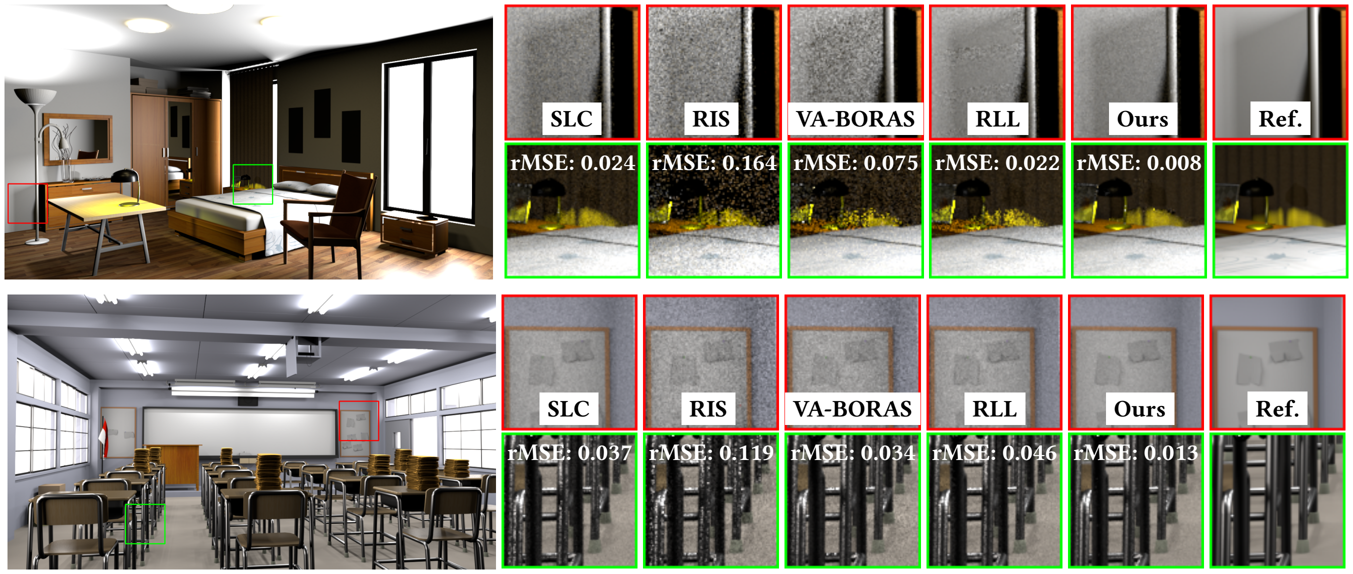


Fig. 4. Bedroom and Classroom. Equal-time comparison (120s) with SLC [Yuksel 2019], RIS [Bitterli et al. 2020; Talbot et al. 2005], VA-BORAS [Rath et al. 2020; Vévoda et al. 2018], RLL [Pantaleoni 2019] and our method. VA-BORAS does not cluster the lights properly. RLL produces stripe artifacts at the top row owing to its update rule. Our method is robust across different configurations and achieves lower error.

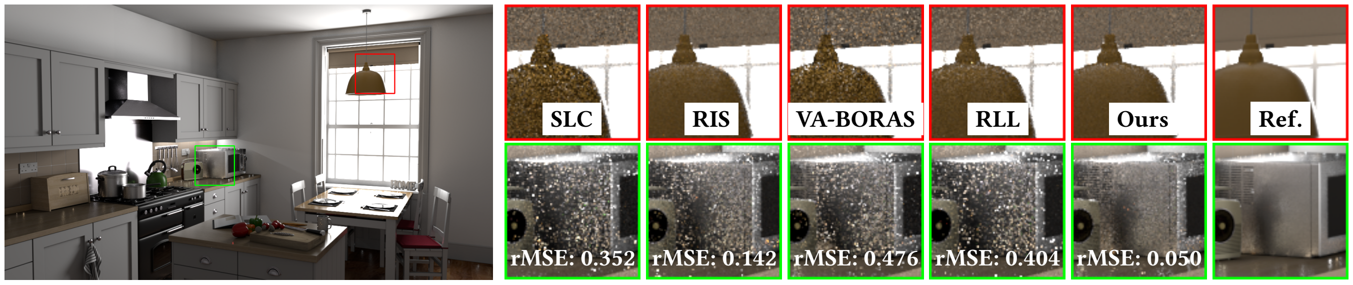


Fig. 5. Kitchen. Equal-time comparison (120s) with 70K virtual point lights on global illumination computation. Our method can handle a large number of lights while significantly outperforming all other methods numerically and visually.

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 combine all methods with BRDF importance sampling using multiple importance sampling [Veach and Guibas 1995] in light contribution estimation *after* each method chooses a light out of the light hierarchy. For BORAS, VA-BORAS, RLL, and our method, we use the same bounding-volume-hierarchy-based scene partition to group the shading points (Section 4.1). 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 90K. 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 α_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 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 on five scenes. We found that, for our method, the better strategy is scene-dependent. We hypothesize that the magnitude of the contribution can lead to different convergence, but we leave further investigation as future work. In Fig. 6, we show the rendering results of the Parking-Lot scene. All methods are combined with BRDF importance sampling before light selection. Our method reduces the relative mean square error to an order of magnitude compared to other methods.

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.

Scene	before light selection	after light selection
Bathroom	0.021	**0.013**
Bedroom	**0.008**	**0.008**
Classroom	**0.012**	0.013
Parking-Lot	**0.047**	0.079
Staircase2	0.004	**0.003**

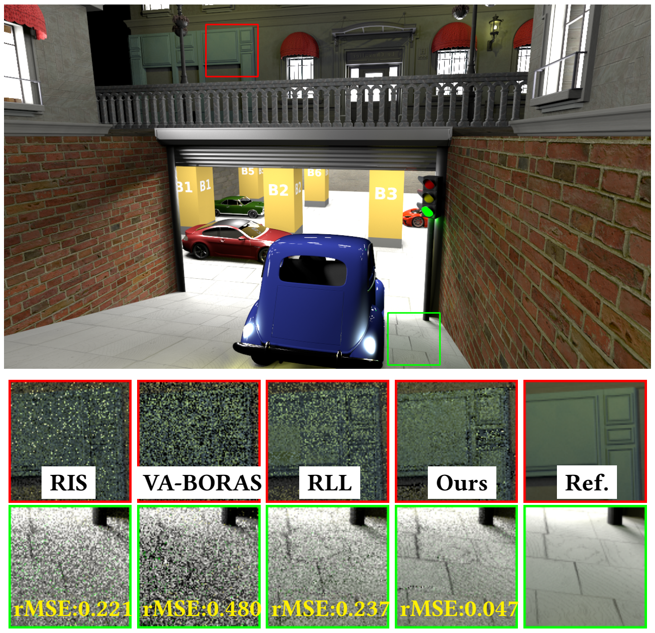


Fig. 6. Parking-Lot. Equal-time comparison (360s) with RIS [Bitterli et al. 2020; Talbot et al. 2005], VA-BORAS [Rath et al. 2020], RLL [Pantaleoni 2019] and our method combining with BRDF importance sampling before selecting a light. Our method significantly outperforms other methods in both visual quality and relative mean square error (rMSE).

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 [Vévoda 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

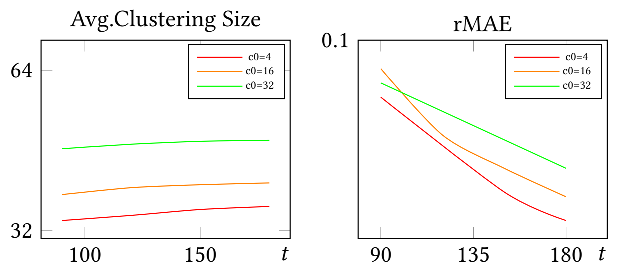


Fig. 7. We compare the evolution of average clustering size and relative mean absolute error (rMAE) for our method with different initial cut sizes using the Bathroom scene. In the experiment, we constrain the maximum cut size to be 64. With a smaller initial cut size, our method achieves a lower error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

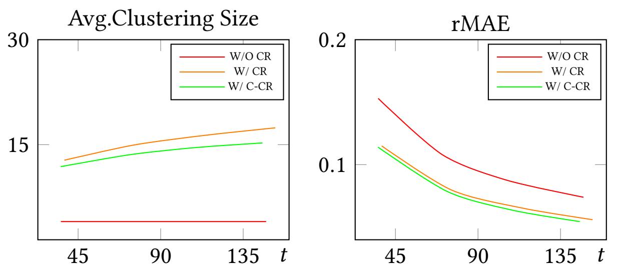


Fig. 8. We compare the evolution of average clustering size, and relative mean absolute error (rMAE) for our method without cluster refinement (W/O CR), with cluster refinement (W/ CR), and with the constrained cluster refinement (W/ C-CR), where we stop splitting when there is no refinement for a while 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 $|C_0|= 4$. We set the maximum cut size as 32 and Γ = 128. The cluster refinement indeed leads to lower error, while the constraints avoid unnecessary clustering and also deliver slightly lower error.

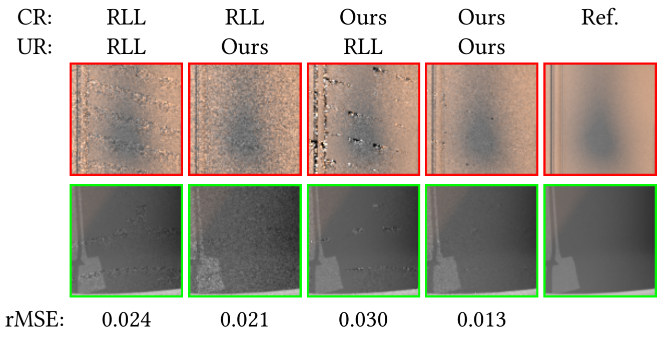


Fig. 9. Ablation study with different combinations of clustering rule (CR) and update rule (UR) of reinforcement lightcuts learning [Pantaleoni 2019] and our method. The relative MSE (rMSE) of each combination is shown at the bottom of the image. The combination of our CR and our UR leads to the fewest image artifacts and the lowest rMSE.

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.

Scene	RLL/RLL	RLL/Ours	Ours/RLL	Ours/Ours
Bathroom	0.024	0.021	0.030	**0.013**
Bedroom	0.022	0.062	0.009	**0.008**
Classroom	0.046	0.034	0.015	**0.013**
Kitchen	0.404	0.491	0.091	**0.050**
Living-Room	0.006	**0.005**	0.018	**0.005**
Parking-Lot	0.237	0.076	0.232	**0.047**
SIA-Shelf	0.042	0.113	0.151	**0.038**
SanMiguel	3.819	2.407	1.099	**0.435**
Staircase	0.013	0.009	**0.007**	**0.007**
Staircase2	0.006	0.005	0.004	**0.003**

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 lighcuts 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/RLL'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.

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.

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 sportscar in Fig. 6 and the SANMIGUEL 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 (Benianus3D), desk (abciuppa), ground lamp (hristopopov), 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

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (2020), 148–1.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *SIAM Rev.* 60, 2 (2018), 223–311.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable realistic rendering with many-light methods. *Computer Graphics Forum* 33, 1 (2014), 88–104.

Carsten Dachsbacher and Marc Stamminger. 2005. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games.* 203–231.

Ken Dahm and Alexander Keller. 2017. Learning Light Transport the Reinforced Way. In *ACM SIGGRAPH 2017 Talks.* Association for Computing Machinery, Article 73, 2 pages.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, Philipp Slusallek, and Kavita Bala. 2010. Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 29, 6 (2010), 1–8.

Michael Donikian, Bruce Walter, Kavita Bala, Sebastian Fernandez, and Donald P Greenberg. 2006. Accurate direct illumination using iterative adaptive sampling. *IEEE Trans. on Visualization and Computer Graphics* 12, 3 (2006), 353–364.

Frédo Durand, George Drettakis, and Claude Puech. 1997. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* 89–100.

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *ACM Comput. Graph. Interact. Tech. (Proc. HPG)* 1, 2 (2018), 25:1–25:17.

Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. 2002. Local Illumination Environments for Direct Lighting Acceleration. In *Proceedings of the 13th Eurographics Workshop on Rendering.* Eurographics Association, 7–14.

Iliyan Georgiev, Jaroslav Křivánek, Stefan Popov, and Philipp Slusallek. 2012. Importance Caching for Complex Illumination. *Computer Graphics Forum (Proc. Eurographics)* 31, 2pt3 (May 2012), 701–710.

Pat Hanrahan, David Salzman, and Larry Aupperle. 1991. A Rapid Hierarchical Radiosity Algorithm. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques.* Association for Computing Machinery, 197–206.

Miloš Hašan, Edgar Velázquez-Armendáriz, Fabio Pellacini, and Kavita Bala. 2008. Tensor clustering for rendering many-light animations. 27, 4 (2008), 1105–1114.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix Row-Column Sampling for the Many-Light Problem. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26, 3 (July 2007), 26–es.

Yuchi Huo, Rui Wang, Shihao Jin, Xinguo Liu, and Hujun Bao. 2015. A matrix sampling-and-recovery approach for many-lights rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 34, 6 (2015), 1–12.

Yuchi Huo, Rui Wang, Ruzahng Zheng, Hualin Xu, Hujun Bao, and Sung-Eui Yoon. 2020. Adaptive Incident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 1, Article 6 (Jan. 2020), 17 pages.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques.* 143–150.

Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* 49–56.

Alexander Keller. 2001. Hierarchical Monte Carlo image synthesis. *Mathematics and Computers in Simulation* 55, 1-3 (2001), 79–92.

Arjan JF Kok and Frederik W Jansen. 1994. Source selection for the direct lighting computation in global illumination. In *Photorealistic Rendering in Computer Graphics.* Springer, 75–82.

Thomas Kollig and Alexander Keller. 2006. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004.* Springer, 245–257.

Eric P Lafortune and Yves D Willems. 1995. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Proceedings of the 6th Eurographics Workshop on Rendering.* Springer, 11–20.

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1 (2020), 1–18.

Yifan Liu, Kun Xu, and Ling-Qi Yan. 2019. Adaptive BRDF-oriented multiple importance sampling of many lights. 38, 4 (2019), 123–133.

Amazon Lumberyard. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/amazon-lumberyard-bistro.

Pierre Moreau, Matt Pharr, and Petrik Clarberg. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing.. In *High Performance Graphics (Short Papers).* 21–26.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Computer Graphics Forum (Proc. EGSR)* 36, 4 (June 2017), 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM Trans. Graph.* 38, 5 (Oct. 2019), 145:1–145:19.

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: matrix slice sampling for the many-lights problem. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 30, 6 (2011), 179:1–179:8.

Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. 2009. Adaptive Wavelet Rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (Dec. 2009), 1–12.

Jacopo Pantaleoni. 2019. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. *arXiv preprint arXiv:1911.10217* (2019).

Jacopo Pantaleoni. 2020. Online path sampling control with progressive spatio-temporal filtering. *SN Computer Science* 1, 5 (2020), 1–16.

Eric Paquette, Pierre Poulin, and George Drettakis. 1998. A light hierarchy for fast rendering of scenes with many lights. 17, 3 (1998), 63–74.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 1266 pages.

Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. 2015. Probabilistic Connections for Bidirectional Path Tracing. *Computer Graphics Forum (Proc. EGSR)* 34, 4 (July 2015), 75–86.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-Aware Path Guiding. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4 (July 2020), 151:1–151:12.

Tobias Ritschel, Thorsten Grosch, Min H Kim, H-P Seidel, Carsten Dachsbacher, and Jan Kautz. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 27, 5 (2008), 1–8.

Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *Ann. Math. Statist.* 22, 3 (1951), 400–407.

Donald B Rubin. 1987. Comment on "The calculation of posterior distributions by data augumentation" by MA Tanner and WH Wong. *J. Amer. Statist. Assoc.* 82 (1987), 543–546.

B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. 2006. Bidirectional Instant Radiosity. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques.*

Eurographics Association, 389–397.

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Trans. Graph.* 15, 1 (Jan. 1996), 1–36.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning.* Vol. 135. MIT press Cambridge.

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Proceedings of the 16th Eurographics Symposium on Rendering.* The Eurographics Association.

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.* Association for Computing Machinery, 419–428.

Edgar Velázquez-Armendáriz, Shuang Zhao, Miloš Hašan, Bruce Walter, and Kavita Bala. 2009. Automatic bounding of programmable shaders for efficient global illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009).

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 4 (2018), 125:1–125:12.

Jirí Vorba, Ondrej Karlík, Martin Sik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4 (2014), 101:1–101:11.

Bruce Walter. 2005. Notes on the Ward BRDF. *Program of Computer Graphics, Cornell University, Technical report PCG-05* 6 (2005).

Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. 2006. Multidimensional Lightcuts. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25, 3 (July 2006), 1081–1088.

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. *ACM Trans. Graph. (Proc. SIGGRAPH)* 24, 3 (2005), 1098–1107.

Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional lightcuts. *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 1–11.

Gregory J Ward. 1994. Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics.* Springer, 11–20.

Yu-Ting Wu and Yung-Yu Chuang. 2013. VisibilityCluster: Average directional visibility for many-light rendering. *IEEE Trans. on Visualization and Computer Graphics* 19, 9 (2013), 1566–1578.

Yu-Ting Wu, Tzu-Mao Li, Yu-Hsun Lin, and Yung-Yu Chuang. 2015. Dual-matrix sampling for scalable translucent material rendering. *IEEE Trans. on Visualization and Computer Graphics* 21, 3 (2015), 363–374.

Cem Yuksel. 2019. Stochastic Lightcuts. In *High-Performance Graphics.* The Eurographics Association, 27–32.

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 on 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 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:

$$L_u(x,c) = \frac{G_u(x,c)M_u(x,c)I_c}{\Lambda(x,c)}, \tag{12}$$

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 [Velázquez-Armendáriz et al. 2009; Walter et al. 2012]), I_c is the total intensity of lights inside the cluster, and $\Lambda(x,c)$ is the *attenuation term*:

$$\Lambda(x,c) = \begin{cases} \frac{1}{d^{\min}(x,c)^2} & \text{if } d^{\min}(x,c) > l_c \;\&\; d^{\min}(x,c') > l_{c'} \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

where $d^{\min}(x,c)$ 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 θ^* of a nondecreasing function f under a zero-mean noise ϵ_n with both bounded mean and variance, if $f'(\theta^*) > 0$, $\sum_{t=1}^{\infty} \alpha_t = \infty$ and $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$:

$$\theta_{t+1} = \theta_t - \alpha_t(f(\theta_t) + \epsilon_n). \tag{14}$$

If we set $\theta_t = Q_t^x(c)$ 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 $\sum_c |Q^x(c) - F_c(x)|$ 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 ϵ_n [Bottou et al. 2018].

C INITIALIZATION AFTER SPLITTING A CLUSTER

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

$$Q_t^x(c) = \left(\prod_{i=1}^{t}(1-\alpha_i)\right) L_u(x,c) + \sum_{j=1}^{t} \alpha_j \left(\prod_{i=j+1}^{t}(1-\alpha_i)\right) \langle F_c(x)\rangle_j. \tag{15}$$

Our learning rate is a monotonically decreasing sequence $\alpha_t = \frac{1}{\beta t^{\omega}}$. We approximate the product as $\prod_{i=1}^{t}(1-\alpha_i) \approx (1-\alpha_t)^t$. Equation (15) is then approximated as:

$$Q_t^x(c) \approx (1-\alpha_t)^t L_u(x,c) + \left(1-(1-\alpha_t)^t\right) \langle F_c(x)\rangle \tag{16}$$

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 $n_{c_1} \approx \frac{L_u(x,c_1)}{L_u(x,c_1)+L_u(x,c_2)} n_c$, where n_c is the visit count of its parent. The initialized Q value is then:

$$Q_t^x(c_1) \leftarrow (1-\alpha_t)^{n_{c_1}} L_u(x,c) + \left(1-(1-\alpha_t)^{n_{c_1}}\right) Q_t^x(c). \tag{17}$$