A Resampled Tree for Many Lights Rendering (supplementary)

Alejandro Conty Estevez
aconty@imageworks.com
Sony Pictures Imageworks
Canada

Chris Hellmuth
chellmuth@imageworks.com
Sony Pictures Imageworks
Canada

Pascal Lecocq
plecocq@imageworks.com
Sony Pictures Imageworks
Canada

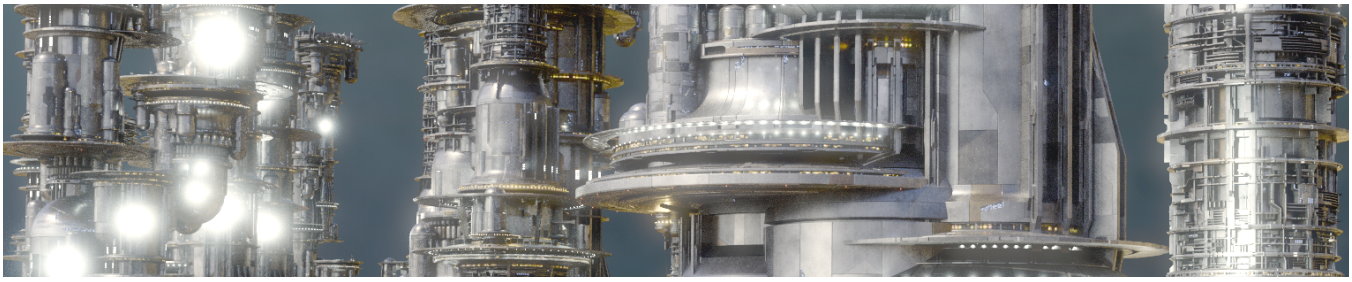


Figure 1: A 100k lights scene from Ant-Man © Marvel Studios. Our new system is able to render specular highlights better than a pure tree-based approach thanks to resampling.

ABSTRACT

We propose a new hybrid method for efficiently sampling many lights on a scene that combines a simplified spatial tree with a resampling stage. Building on previous methods that work with a split or cut of the light tree, we introduce the idea of probabilistic splitting to eliminate noise boundaries. This yields a small but unbounded subset of lights that is then reduced to a smaller and bounded set which is used for full light/BSDF evaluation for resampling. Our main contribution is the stochastic splitting formulation combined with a Reservoir Set concept which can limit samples to an arbitrary number, allowing two stages of resampling with different heuristics.

CCS CONCEPTS

• **Computing methodologies** → **Rendering**; **Ray tracing**.

KEYWORDS

: illumination, ray tracing, many lights

ACM Reference Format:
Alejandro Conty Estevez, Chris Hellmuth, and Pascal Lecocq. 2024. A Resampled Tree for Many Lights Rendering (supplementary). In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH Talks '24), July 27 - August 01, 2024.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641233.3664352

1 INTRODUCTION

Rendering with many lights – on the order of millions or even thousands – is a challenging problem. Efficiently choosing the relevant emitters to a shading point involves finding a balance between performance and variance. We seek to render with as few shadow rays per pixel as possible. It is therefore key to design a good importance sampling technique.

Ideally we would use a perfect scheme where the probability of choosing a light is linearly proportional to the BSDF, light power, geometric decay and occlusion. But this is not always possible in an offline render with complex scenes and shading.

2 PREVIOUS WORK

Many light sampling techniques have gained attractiveness in recent years, mainly driven by novel light tree-based solutions and new techniques tailored for real-time ray tracing on the GPU.

Most of the light-tree solutions share many similarities with the Lightcuts technique introduced by Walter et al. [Walter et al. 2005] in the context of many Virtual Point Lights. That is, a binary space partition clusters the lights in a hierarchy, where each cluster approximates the lighting contained in the sub-tree. Given a shading point, selecting the lights that contribute the most to the surface illumination is done by walking down the tree, discarding sub-trees with less contribution. However, finding a metric that represents a good cluster importance is challenging and depends heavily on the quality of the tree construction but also on the tree traversal decisions.

In our previous work [Conty Estevez and Kulla 2018], we organize millions of heterogeneous lights into a BVH tree built using a modified surface heuristic region that better regroups lights with similar orientation and emission profile. The tree is stochastically traversed based on an importance measure from an approximate lighting contribution. An Adaptive Tree Splitting strategy (ATS) based on a quality score is used to force the exploration of multiple branches to mitigate the initial poor importance estimation. A GPU implementation has been proposed by [Moreau and Clarberg 2019] and further improved using a 2-levels BVH for faster tree rebuild in the context of dynamic illumination [Moreau et al. 2022]. The solution improves the sampling quality by an order of magnitude

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06.
https://doi.org/10.1145/3641233.3664352

and also support participating media. However, the method may introduce sampling discontinuities due to the deterministic choice of splitting that depends on the spatial location of the shading point.

Stochastic Lightcuts [Lin and Yuksel 2020; Yuksel 2020] addresses the correlation problem arising from the use of a predetermined representative light in the original Lightcuts hierarchy. Instead, the representative light is chosen stochastically by traversing the tree down to a leaf, similar to [Conty Estevez and Kulla 2018]. However, performing a tree traversal for each visited cluster node can be computationally expensive. To minimize the cost, cuts are shared across the image space and interleaved to hide any correlation artifacts. However, the cost saving is limited to the primary hits and do not account for the presence of participating media.

Machine learning based methods [Pantaleoni 2019; Vévoda et al. 2018; Wang et al. 2021] improve the quality of Lightcuts by learning the visibility of light samples from the previous lighting results. These methods usually require a 3D grid data structure to cache and update the cut probabilities from the learned visibility. However, maintaining such a data structure is challenging because our production scenes can be massively complex, with highly detailed surfaces at multiple scales, as shown in Fig 1. In addition, participating media rendering with many lights is not addressed in these techniques.

Resampled Importance Sampling (RIS) [Talbot 2005; Talbot et al. 2005] propose an interesting set of Monte Carlo estimators that have been used in the context of real-time raytracing with many lights on the GPU by Bitterili et al [Bitterli et al. 2020]. Using a spatio-reservoir sampling technique (ReSTIR), an approximate initial set of light samples is selected from an energy-based importance measure and successively resampled locally, spatially, and temporally. The end result is a well-selected light sample that better represents the sample product of the incident light with the bsdf. A key element of the technique is the use of a weighted reservoir sampler [CHAO 1982], which operates in linear time on the input without prior knowledge of the candidates importance. This approach avoids the costly construction and storage of CDF tables, making the solution tailored for GPU applications. Boksansky et al [Boksansky et al. 2021] later extended the solution to handle secondary rays in world space using a 3D grid, but with the aforementioned limitation of such data structure. Many-lights in the presence of participating media is not also addressed.

Our solution leverages Adaptive Tree Splitting [Conty Estevez and Kulla 2018] with a reservoir resampling approach in the spirit of [Bitterli et al. 2020]. We propose a novel stochastic splitting strategy, which both simplifies the tree structure and removes the sampling discontinuities seen in our previous work. We use a reservoir resampling solution to select a small bounded set of good light sources from the unbounded set obtained during the light tree traversal. We introduce a new reservoir set that behaves like a n-size weighted reservoir sampler with replacement, but without duplicates. In a second step, we generate samples from the selected light sources and use the extensive BSDF evaluation and resampling to retain well chosen light samples. Our solution is GPU friendly as we operate in a streamline fashion using reservoir samplers with a minimal storage cost. It is not limited to primary bounces and naturally enables the support of participating media.

3 OVERVIEW OF OUR SAMPLING

In our previous research [Conty Estevez and Kulla 2018], we employed a complex sampling tree that integrated both spatial and emitter orientation partitioning. This design enabled us to establish an importance measure that considered the orientation of the light and its impact on the shading point. Although beneficial, this approach incurred a higher traversal cost. However, this cost was justified as the traversal process was solely accountable for generating the final samples. In our current approach, we have opted to streamline this traversal process for enhanced speed. Instead of relying entirely on the traversal for sample quality, we now defer the refinement of the samples to a subsequent resampling stage. This shift not only accelerates the traversal process but also improves the quality of the final output through effective resampling.

The tree structure comprises clusters of emitters, denoted as $C_1, C_2, \ldots, C_n$ (representing interior nodes), and individual single emitters, represented as $E_1, E_2, \ldots, E_n$ (constituting the leaves). We employ a tree importance heuristic, H, to guide the traversal. This heuristic is exclusively spatial when evaluating clusters, and we will refer to this relaxed version as $h()$, but it will incorporate directionality when assessing individual emitters, $H()$. This aspect of the heuristic proves particularly effective and precise for flat and spot light emitters. Consequently, the sampling pipeline can be summarized as follows:

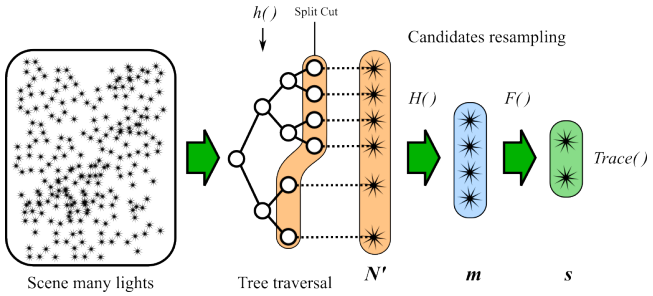


Figure 2: Pipeline of the sampling process from the scene lights to shadow rays.

When employing tree traversal with splitting, we acquire a limited yet unbounded set of candidate lights, guided by the tree's importance heuristic. Notably, this heuristic tends to be less effective for larger clusters near the top of the tree, resulting in a mixture of both suitable and unsuitable candidates. The effectiveness of these candidates is more accurately assessed using the cost-efficient heuristic at the tree's leaves (the actual emitters). This assessment forms the basis of our initial resampling process, which utilizes a reservoir set. By employing the same unnormalized importance value derived from the tree, we achieve two key objectives:

- We limit the number of candidates to a manageable set, facilitating a more comprehensive and resource-intensive resampling process.
- We effectively filter out the less desirable candidates that are often mistakenly prioritized by the cluster importance heuristic during traversal.

By combining one light traversal and two resampling stages we go from many N lights in the scene to N' unbounded candidate

Symbol	Meaning	Example
N	total lights in the scene	many
$N' \leq N$	candidates from the split tree traversal	$1-60$
$m \leq N'$	reduced candidates by resampling with H	16
$s \leq m$	reduced shadow rays by resampling with F	$1-4$

Table 1: Semantics of the number used for our tree sampling

lights, then narrow down to m user controlled candidates and finally s shadow rays. Satisfying $s \leq m \leq N' \leq N$ we can summarize the semantics of the numbers as shown in Table 1

Following the tree traversal, we obtain a set of N' light candidates denoted as L_i, each associated with a probability $P_{tree}(L_i)$. This probability is well-defined, normalized, and deterministic. Up until this stage of the process, a simplified heuristic $h(C)$ is utilized for interior nodes (clusters), and a better, orientation aware, heuristic $H(L_i)$ for the leaf emitters. The resampling weight W_h comes from the same heuristic importance

$$I_h(L_i) = \frac{H(L_i)}{P_{tree}(L_i)}$$

However, once we achieve a bounded subset of m lights, the focus shifts to sampling the candidate lights L_i. At this point, we conduct a comprehensive evaluation of the Bidirectional Scattering Distribution Function (BSDF) and the irradiance E of a random light sample S_i from light L_i. Subsequently, the final reservoir sampling is based on the outcomes of this full evaluation

$$F(S_i) = f(\omega_i, \omega_o)\frac{E(S_i)}{P_{tree}(L_i)P(S_i/L_i)}W_{mis}W_h(L_i) \qquad (1)$$

where the function f denotes the BSDF at the given surface, and W_{mis} refers to the Multiple Importance Sampling (MIS) weight, which ignores resampling probabilities for correctness. The second resampling pass operates on importance $I_f(S_i) = F(S_i)$ yielding a final weight $W_f(S_i)$. This process effectively narrows down the sample pool for generating the final s shadow rays. While typically a single shadow ray is sufficient, scenes characterized by significant shadow variance stand to gain from the use of multiple shadow rays. The final contribution of the sample S_i is $F(S_i)\ W_f(S_i)$ and boils down to

$$f(\omega_i(S_i), \omega_o)\ \frac{E(S_i)}{P_{tree}(L_i)P(S_i/L_i)}\ W_{mis}\ W_h(L_i)\ W_f(S_i)$$

Which is the usual Monte Carlo weight with the addition of W_h and W_f from the resampling. In summary, our sampling will call:

- Inexpensive heuristic $h()$ many times during traversal,
- Orientation aware heuristic $H()$ for leaves and first resampling,
- Full light/BSDF evaluation m times for the second resampling and
- Full shadow ray trace s times,

where m and s are render parameters. The former controls the quality of the chosen samples and the latter how many shadow rays we want to trace per shading point.

4 RESERVOIR SET

A weighted reservoir sampler usually takes an unlimited number of items or candidate samples and selects a random one based on an unnormalized importance measure. For our system to work we needed to extend the concept to produce more than just one result from the inputs. That is, from an unlimited input set, select a random subset of n distinct samples chosen according to the same importance measure.

To accomplish this, we create n independent weighted reservoir samplers, each containing a single item, and distribute the input candidates among them randomly. And to avoid clumping, we perform this shuffling with a random permutation whose seed changes every n input candidates.

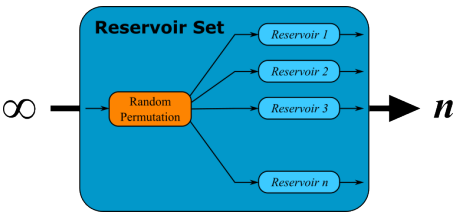


Figure 3: Reservoir Set

The result is that every reservoir in the set sees a random subsequence of the input stream. This randomization strategy effectively eliminates any bias in the order of the input candidates. Each of the n reservoirs selects a single candidate, resulting in a pool of n or fewer candidates, each one chosen according to the importance measure. Additionally, this method satisfies several noteworthy conditions:

- After processing $k \times n$ candidates, each reservoir has seen exactly k of them.
- Every candidate is allocated to exactly one reservoir, ensuring that there are no duplicates in the output.
- When $k \geq n$, all reservoirs will be filled. Conversely, if $k \leq n$, the output will precisely match the input set.

The proof that our reservoir set is unbiased is left to be investigated, but we found connections with the Stratified Resampled Importance Sampling method described by Talbot in his master thesis [Talbot 2005].

Building upon this approach, we implement the operation

$$\text{ReservoirSet}(U, f, n) \rightarrow L.$$

This function takes an input set U of unlimited size and outputs a set L consisting of n or fewer candidates, each selected based on f, the designated importance measure.

5 LIGHT HIERARCHY CONSTRUCTION

In our approach, we adopt the light emitter tree structure from the preceding paper but introduce a significant simplification: clusters of light emitters no longer maintain orientation bounds. We observed that when various types of light emitters are grouped together, the combined orientation bounds often become irrelevant.

Therefore, we now store these bounds exclusively at the leaf emitters of the tree, where they effectively limit a light's influence, as originally intended.

This modification streamlines the traversal process in the interior nodes of the tree, enhancing its efficiency. During traversal, we retain solely the spatial component of the importance heuristic from our prior work, up until we encounter a leaf emitter. It is at this stage that we include the orientation factor in our evaluation. We can, for simplicity, assume a binary tree, although the actual topology is an implementation detail. Regardless of this, sampling in the tree is a random walk where we use the heuristic to choose a path from the root to the emitters.

Ultimately, a single traversal of our unsplit tree results in the selection of exactly one light emitter. However, when cluster nodes are split, the traversal produces several light emitters. These N' emitters are then utilized in the subsequent resampling phase, a key advancement in our methodology described in the previous section.

6 CLUSTER AND LEAF EMITTER IMPORTANCE

Our clusters of lights are simply defined by a bounding sphere and total contained energy. We simplified our heuristic to ignore any orientation of the emitters inside, and we follow an inverse square distance decay limited by the bounds. That is, given a cluster, C, with center C_c, radius C_r, and energy C_e we define

$$h(C) = \frac{I(C,P)\ \sqrt{C_e}}{\max(C_r,\ |C_c - p|)^2},$$

where p is the shading point, and $I(C,p)$ is a conservative irradiance estimation from the cluster to the shading point. This irradiance estimation is computed from the angle of the $C_c - p$ vector with the surface normal θ_n and the angle θ_c subtended by the bounding sphere at the shading point. Then

$$I(C,p) = \cos(\max(0, \theta_n - \theta_c)),$$

completes the definition. Note, we are using the square root of the energy $\sqrt{C_e}$, which was empirically found to work better than the physical estimation to avoid sampling starvation due to large clusters where the heuristic is very inaccurate.

For the leaves of the tree representing a single emitter or light, L, we do use the orientation bounds as in our previous work. This is especially useful with spotlights, where the light is contained within a small cone. We summarize this logic in a coverage function $\mathrm{Cov}(L,p)$ that returns a value in $[0,1]$ to clip the energy. Then, at the leaf level, a more accurate heuristic is given by

$$H(L) = \frac{\mathrm{Cov}(L,P)\ I(L,p)\ L_e}{\max(L_r,\ |L_c - p|)^2}.$$

Here, we give more confidence to the light energy L_e by forgoing the square root, and we trust the orientation bounds to be more meaningful via the coverage function.

7 LIGHT PROBABILITY FROM TRAVERSAL

In the absence of splitting, the probability of selecting a light cluster C from its parent $C\uparrow$ in the tree, based on importance, is denoted as $P_i(C|C\uparrow)$. The probability of reaching cluster C can be recursively defined as follows:

$$P_{nosplit}(C) = P_i(C|C\uparrow)P_{nosplit}(C\uparrow)$$

This calculation continues up to the root of the tree. However, when splitting is introduced, we must consider a new event, $s(C)$, which represents the splitting of the cluster. This involves traversing all of its children instead of selecting one randomly. The probability of splitting, denoted as $P(s(C))$ and simplified here as $P_s(C)$, is constrained to decrease as we descend the tree:

$$P_s(C) \le P_s(C\uparrow)$$

Additionally, we set the splitting across the entire tree using the same random number for a given hit point on the geometry, without any warping. Logically this translates to $s(C) \to s(C\uparrow)$, meaning that splitting a cluster implies that all of its predecessors are also split. The probability of not splitting, denoted as $P_{\hat{s}}(C)$, is simply defined as $1 - P_s(C)$. Thus, the probability of our split traversal reaching a cluster is given by:

$$P(C) = P_s(C\uparrow)\cdot P(C\uparrow|s(C\uparrow))\cdot 1 + P_{\hat{s}}(C\uparrow)\cdot P(C\uparrow|\hat{s}(C\uparrow))\cdot P_i(C|C\uparrow).$$

But $P(C\uparrow|s(C\uparrow))$ given our split restrictions also equals 1, so the simplified equation is just

$$P(C) = P_s(C\uparrow) + P_{\hat{s}}(C\uparrow)\cdot P(C\uparrow|\hat{s}(C\uparrow))\cdot P_i(C|C\uparrow). \quad (2)$$

Then we just need to define the conditional probability for reaching a cluster knowing that it has not been split $P(C|\hat{s}(C))$, which has another recursive definition:

$$P(C|\hat{s}(C)) = P_s(C\uparrow|\hat{s}(C)) + P_{\hat{s}}(C\uparrow|\hat{s}(C))\cdot P(C\uparrow|\hat{s}(C\uparrow))\cdot P_i(C|C\uparrow).$$

Here,

$$P_s(C\uparrow|\hat{s}(C)) = \frac{P_s(C\uparrow) - P_s(C)}{1 - P_s(C)}$$

represents the probability of splitting the parent of cluster $C\uparrow$ given that C has not been split. Conversely, $P_{\hat{s}}(C\uparrow|\hat{s}(C))$ is the complement to 1.

With these equations, we can calculate probabilities for any cluster in the tree, though our primary interest lies in the leaves. Nevertheless, we can simplify the mathematics to aid the implementation. To this end, we will compute two quantities for every cluster as we traverse down the tree: $P_s(C)$, which we already know, and $T(C)$, defined as:

$$T(C) = \begin{cases} 1 & \text{if } C = \text{root} \\ P_i(C|C\uparrow)\cdot(P_{ns}(C) + (1 - P_{ns}(C))\cdot T(C\uparrow)) & \text{otherwise} \end{cases} \quad (3)$$

Here, $P_{ns}(C) = P_s(C\uparrow|\hat{s}(C))$ is used for brevity. When we stop at a cluster in the tree, we can easily compute the probability of reaching it as:

$$P(C) = P_s(C\uparrow) + (1 - P_s(C\uparrow))\cdot T(C). \quad (4)$$

This approach streamlines the process of determining the probability of reaching a specific light in the tree while dealing with the complexities of cluster splitting.

In Algorithm 1 we provide a recursive implementation that calls a *Visit*() procedure when it gets to a leaf emitter. This procedure would be responsible for the first resampling stage. To begin the traversal, one would just call *Traverse*(*root*, 1, ξ_s, ξ_t), where ξ_s is the

Algorithm 1 Stochastic splitting tree traversal

procedure IMPORTANCE(C)
  return $H(C)$ **if** C *is leaf,* **otherwise** $h(C)$
end procedure
procedure TRAVERSE(C, T, ξ_s, ξ_t)
  if C *is leaf* **then**
    $PDF \leftarrow P_s(C\uparrow) + (1 - P_s(C\uparrow)) \cdot T$
    VISIT(C, PDF)
  else
    $I_l, I_r \leftarrow$ IMPORTANCE(C_l), IMPORTANCE(C_r)
    $split \leftarrow \xi_s < P_s(C)$
    $P_l \leftarrow 1$ **if** *split,* **otherwise** $\frac{I_l}{I_l+I_r}$
    $P_r \leftarrow 1$ **if** *split,* **otherwise** $1 - P_l$
    $\xi_t' \leftarrow \xi_t$ **if** *split,* **otherwise** $warp(\xi_t, P_l, P_r)$
    ▷ Probability of splitting $C\uparrow$ if C is not split
    $S_{ns} \leftarrow (P_s(C\uparrow) - P_s(C)) / (1 - P_s(C))$
    ▷ Probability of C if C is not split
    $P_{ns} \leftarrow S_{ns} + (1 - S_{ns}) \cdot T$
    if $\xi_t < P_l$ **then**
      TRAVERSE($C_l, P_{ns} \cdot P_l, \xi_s, \xi_t'$)
    end if
    if $\xi_t \geq P_l$ **then**
      TRAVERSE($C_r, P_{ns} \cdot P_r, \xi_s, \xi_t'$)
    end if
  end if
end procedure

random number for the split or cut, and ξ_t is the random number for the rest of the traversal. Note a single random number would also be possible by warping when splitting stops, but this makes the code cleaner.

8 STOCHASTIC SPLITTING

We consider as problematic those clusters whose bounding radius is large compared to the distance from the cluster center to the shading point. The simplest possible heuristic for splitting a cluster is the one that, given the shading point p, assigns a splitting probability $P_s(C)$ such as

$$P_s(C) = \begin{cases} 1 & \text{if } |C_c - p| < C_r \\ 0 & \text{otherwise} \end{cases}.$$

But using a smooth function like a Cauchy bell shaped curve like

$$P_s(C) = \frac{1}{1 + t^2/\gamma^2} \quad where \quad t = \frac{\max(|C_c - p| - C_r, 0)}{C_r} \tag{5}$$

produces a smoother picture and introduces a render parameter γ to control how aggressive splitting is. The greater the γ value, the more candidates the traversal will feed into the resampling. Note this curve satisfies our constraint $P_s(C) \leq P_s(C\uparrow)$ since the radius C_r always decreases as we walk down the cluster tree. It is also possible to measure t from the center of the cluster instead of the edge as in our proposal. This curve does not satisfy the constraint, but our implementation enforces it by clipping the curves. The result is still useful for experimenting with lower splitting rates.

9 PARAMETERIZATION OF THE SAMPLING AND RESULTS

We deployed our new light tree algorithm to production last year, and have observed decreases in noise and render times on our production scenes. The new algorithm is also implemented on GPU alongside the CPU, where we benefit from our s parameter's effect of tracing the same number of shadow rays for each sample. This helps to reduce our thread divergence and increase our rendering throughput. In this section, we evaluate our method on three test scenes, comparing against our previous implementation and analyzing optimal m- and γ-parameter values.

Figure 4: Cityscape test scene rendered at 65,000 spp

Cityscape. Our cityscape scene is a procedurally-built skyline meant to test the scaling capabilities of our light tree. Using only simple shaders and shapes, the scene helps isolate the costs of light sampling. We compare against our previous method with a tree of 10,000,000 lights. We observe a small increase in variance per sample, but our rendering times are improved sufficiently so that the new method is more efficient, as seen in Fig 5.

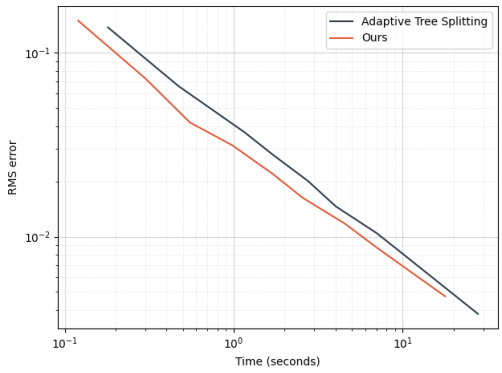


Figure 5: Error comparison of our method against our previous work on the cityscape scene featuring over 10,000,000 lights. The previous method has slightly lower variance per sample due to more sophisticated light evaluations, but our method is more efficient.

Chronopolis. We also tested on the production Chronopolis environment, which features 52,000 spot lights, 27,000 area lights, a skydome and a directional light. The towers in the environment are assembled from over 700,000 instances and shaded with production OSL networks. With high overhead from shading and ray-tracing, our new implementation is only marginally faster than our previous

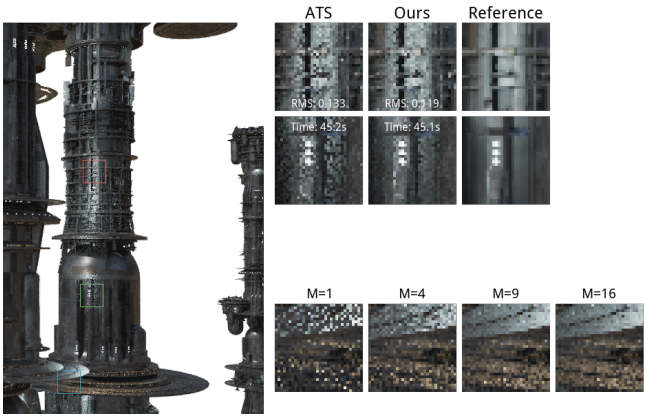


Figure 6: Chronopolis tower scene featuring over 80,000 lights. The first two insets show an improvement in variance of our method against Adaptive Tree Splitting. Third inset shows how varying reservoir count m improves variance.

technique. Instead we gain efficiency through lower variance, as seen in Fig 6.

We also observe the benefit of increasing reservoir size m on the Chronopolis environment, gaining significant improvements in variance in glossy metallic areas where the BSDF plays a critical role in the contribution of a given light. Due to the complexity of the scene's shaders and geometry, we are able to increase m for very little cost in overall render times.

Participating Media. We have observed in production scenes featuring participating media that our new technique is significantly more efficient than our previous method. We use a production environment to experiment with two measures of split probability functions. With the boundary heuristic, we split inversely proportional to $\frac{\max(|C_c - p| - C_r, 0)}{C_r}$, as specified in Section 8. The simpler center heuristic splits inversely proportional to $\frac{|p - C_r|}{C_r}$. Fig 7b compares the two heuristics, and Fig 7a analyzes the convergence of a sweep across different γ values and the two approaches, finding that splitting aggressively is the most efficient strategy in participating media.

REFERENCES

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.* 39, 4, Article 148 (aug 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. *endering Many Lights with Grid-Based Reservoirs.* Apress, Berkeley, CA, 351–365. https://doi.org/10.1007/978-1-4842-7185-8_23

M. T. CHAO. 1982. A general purpose unequal probability sampling plan. *Biometrika* 69, 3 (12 1982), 653–656. https://doi.org/10.1093/biomet/69.3.653 arXiv:https://academic.oup.com/biomet/article-pdf/69/3/653/591311/69-3-653.pdf

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 25 (aug 2018), 17 pages. https://doi.org/10.1145/3233305

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of I3D 2020)* 3, 1 (2020), 18 pages. https://doi.org/10.1145/3384543

Pierre Moreau and Petrik Clarberg. 2019. Importance Sampling of Many Lights on the GPU. In *Ray Tracing Gems*, Eric Haines and Tomas Akenine-Moller (Eds.). APress, Berkeley, CA, 255–283. https://doi.org/10.1007/978-1-4842-4427-2_18

P. Moreau, M. Pharr, and P. Clarberg. 2022. Dynamic many-light sampling for real-time ray tracing. In *Proceedings of the Conference on High-Performance Graphics* (Strasbourg, France) *(HPG '19)*. Eurographics Association, Goslar, DEU, 21–26. https://doi.org/10.2312/hpg.20191191

Jacopo Pantaleoni. 2019. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. *CoRR* abs/1911.10217 (2019). arXiv:1911.10217 http://arxiv.org/abs/1911.10217

Justin Talbot. 2005. Importance Resampling for Global Illumination. In *Master Thesis*. Birgham Young University. https://scholarsarchive.byu.edu/etd/663/

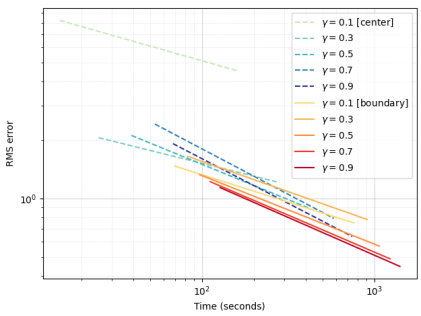
Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering (2005)*, Kavita Bala and Philip Dutre (Eds.). The Eurographics Association. https://doi.org/10.2312/EGWR/EGSR05/139-146

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. *ACM Trans. Graph.* 37, 4, Article 125 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201340

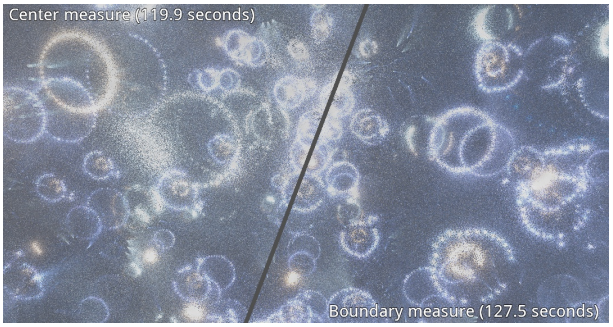
Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph.* 24, 3 (jul 2005), 1098–1107. https://doi.org/10.1145/1073204.1073318

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 (dec 2021), 10 pages. https://doi.org/10.1145/3478513.3480561

Cem Yuksel. 2020. Stochastic Lightcuts for Sampling Many Lights. *IEEE Transactions on Visualization and Computer Graphics* (2020), 11 pages. https://doi.org/10.1109/TVCG.2020.3001271



(a) Convergence plot of varying γ parameters. Solid red lines indicate that t in P_s uses the distance to the cluster boundary in its numerator, while dotted blue lines use the distance to the cluster center. More aggressive splitting in the former leads to slower frames, but more efficient convergence.



(b) Approximately equal time comparison of two splitting heuristics with $\gamma = 0.9$. On the left we split in inverse proportion to the distance to the cluster center, and on the right the cluster's boundary. Under the center heuristic we see more regions with high variance.

Figure 7: Participating media results and convergence