Importance Sampling of Many Lights with Adaptive Tree Splitting

ALEJANDRO CONTY ESTEVEZ, Sony Pictures Imageworks
CHRISTOPHER KULLA, Sony Pictures Imageworks

Fig. 1. A procedural city with 363,036 lights, one GI bounce and participating media. Rendered with 16 samples per pixel, each shading point takes an average of 7 shadow rays (45 for the volume integral). We shoot an average of 1700 rays per pixel. The image rendered in 20 minutes on a quad core Intel i7.

We present a technique to importance sample large collections of lights (including mesh lights as collections of small emitters) in the context of Monte-Carlo path tracing. A bounding volume hierarchy over all emitters is traversed at each shading point using a single random number in a way that importance samples their predicted contribution. The tree aggregates energy, spatial and orientation information from the emitters to enable accurate prediction of the effect of a cluster of lights on any given shading point. We further improve the performance of the algorithm by forcing splitting until the importance of a cluster is sufficiently representative of its contents.

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

Additional Key Words and Phrases: illumination, ray tracing, many lights

ACM Reference Format:
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 (August 2018), 17 pages. https://doi.org/10.1145/3233305

1 INTRODUCTION

Direct lighting calculations are a critical part of modern path tracing renderers with next event estimation. While sampling from simple light shapes [Shirley et al. 1996] is well understood, relatively little attention has been devoted to the problem of efficiently sampling from large collections of such shapes. In production renderers, this problem appears both in the form of scenes containing many distinct lights (Figure 1), and scenes with meshes acting as emitters (sometimes

Authors' addresses: Alejandro Conty Estevez, Sony Pictures Imageworks; Christopher Kulla, Sony Pictures Imageworks.

2577-6193/2018/8-ART25 $15.00
https://doi.org/10.1145/3233305

referred to as "mesh lights") (Figure 15) where the triangles making up the surface act as a large collection of emitters. Naturally both cases are often combined as well, for example in a night time city scape illuminated by many intricate neon signs and street lamps. We address these goals by introducing a hierarchy over the lights that is efficient to build (similar in cost to building acceleration structures for ray intersection) as well as an efficient traversal mechanism that allows choosing lights from a large set with sub-linear cost and in a way that importance samples their expected contribution.

The use of hierarchical structures to guide sampling of many lights similarly to our method has been discussed by practitioners in recent years [Gospodnetić 2017; Keller et al. 2017; Vegdahl 2014], though to our knowledge no publication discusses in detail their approach to tree construction or traversal. This paper aims to describe how we solved these problems and makes the following original contributions:

- The *Surface Area Orientation Heuristic* (SAOH) which guides construction of the hierarchy (Section 4.4)
- An *importance measure* from a cluster of lights to a shading point (Section 5.1) or ray segment in participating media (Section 5.2)
- *Adaptive tree splitting* which improves performance by delaying stochastic traversal until certain criteria are met (Section 5.4)

For completeness, we aim to describe all other details of our algorithm as well including exactly how we represent each light cluster (Section 4) and how to efficiently traverse the tree for sampling (Section 5).

2 PREVIOUS WORK

Naively sampling every light source in a scene is impractical above a few dozen light sources. In fact, the mere act of looping over the lights can be quite costly even if shadow rays are skipped. Early optimization techniques such as shadow ray reduction [Ward 1994] are not well-suited to scenes with more than a few hundred light sources because they still require a linear scan across lights.

Early work on this problem [Shirley et al. 1996] recognized that while a uniform or energy based PDF over the lights can produce reasonable results when all lights have similar influence over the image, this quickly breaks down in scenes where the light sources have more localized influence. Building a probability density function (PDF) per point is impractical as it would require the loop over all lights we are trying to avoid. One approach [Vévoda and Křivánek 2016] is to amortize the creation of localized PDFs within cells of a uniform grid, but the initialization of such a structure can be quite costly. We perform a more detailed comparison in Section 7.

Another simple yet effective strategy is to bound the region of influence of each light and insert them into a hierarchy to quickly select which lights affect any given point [Bikker 2007]. Unfortunately this method requires clamping or modifying the light falloff to avoid infinite extents which introduces bias. Stochastic Light Culling [Tokuyoshi and Harada 2016] randomizes the light extent to remove bias, but it requires several passes with different trees to hide the artifacts. In scenes with densely packed light sources (Figure 17) the amount of overlap between lights can still be quite high.

Rendering with many lights has also been studied in the context of VPL rendering. For a detailed overview of this long line of research, we refer the reader to the excellent state of the art report by Dachsbacher et al. [2014]. In this context, light sources are first discretized into a large pool of path vertices, which are then combined with eye paths to form the image. The set of VPLs may be large (sometimes even including indirect lighting path vertices to represent indirect illumination

as well) which has led to the creation of various scalable methods to reduce the cost of looping over all VPLs to sum up the contributions. Hierarchical techniques such as Lightcuts [Walter et al. 2005] or its predecessor [Paquette et al. 1998] avoid this linear cost during rendering by building a hierarchy over the lights upfront and traversing the tree per shading point to evaluate the large sum approximately. The biggest drawback of VPL methods lies in the initial discretization of the illumination into points. This requires rebuilding the tree over different discretizations per pass to remove correlation artifacts and introduces a weak singularity near each VPL which must be clamped or otherwise accounted for. In contrast, our method assumes individual light sources can be importance sampled directly and that the important problem to solve is light *selection*. Likewise, we only concentrate on direct lighting calculations as we assume indirect lighting is accounted for by path tracing. Nonetheless, our *adaptive tree splitting* heuristic bears some resemblance to the *cut* idea introduced in Lightcuts [Walter et al. 2005] which we discuss in more detail in Section 5.4.

Concurrently with our work, Vévoda et al. [2018] have studied the same problem from a different perspective. They use a light hierarchy to produce small clusters of lights and use learning techniques to derive cluster selection probabilities with data gathered during rendering. The data is cached spatially as in their precursor work [Vévoda and Křivánek 2016] to adapt to changing conditions within the image. This allows them to take visibility information into account and therefore avoid spending samples on fully occluded lights. On the other hand, they do not directly address participating media or study in detail the impact of grid resolution on their approach. For instance we believe that the simple heuristic of splitting 64 times along the shortest scene dimension may fail in scenes with wide spatial extents. Their work is largely orthogonal to ours and we discuss opportunities to combine their insights with ours in the conclusion (Section 8).

3 METHOD OVERVIEW

Modern path tracers rely on multiple importance sampling to efficiently combine importance sampling from the light shape and the shading point's BSDF [Veach and Guibas 1995]. We observed that we had to use a bounding volume hierarchy to accelerate intersection tests of *BSDF samples* against large collections of lights. We therefore started exploring ways to use a similar hierarchy for importance sampling the *light samples* as well.

We will present our method in two parts. We first describe the light hierarchy construction (Section 4). Here we will detail how our hierarchy differs from those traditionally used for fast intersection testing. In particular, we will describe the contents of each cluster (Section 4.1), how we take into account the orientation of lights during construction (Section 4.3) and how this drives the heuristic we use for top-down construction (Section 4.4).

Secondly, we describe how this tree is useful during rendering by describing how we can predict the relative importance of a cluster to a given shading point for surfaces (Section 5.1) or ray for participating media (Section 5.2). This importance measure directly drives our stochastic tree traversal which is used for choosing which lights to sample (Section 5.3). Finally we demonstrate our adaptive splitting criteria which delays stochastic tree traversal (Section 5.4) until certain criteria are met to improve quality.

4 LIGHT HIERARCHY CONSTRUCTION

4.1 Cluster Representation

Traditional bounding volume hierarchies (BVHs) as used for intersection testing only store spatial bounding volumes. In our case, we need to augment this with additional information to provide bounds on the cone of normals and emission profiles contained within a cluster.

Table 1. Values for the orientation bound attributes for a few common examples. The angle θ_e does not contribute anything in the Sphere and Point lights, but we define it as $\pi/2$ for consistency.

Light Type	Axis	θ_o	θ_e
Quad	quad's normal	0	$\pi/2$
Sphere	any unit vector	π	$\pi/2$
Point	any unit vector	π	$\pi/2$
Spot	spot's axis	0	spot aperture
Mesh	from root cluster of hierarchy		

We represent the orientation cone with an axis and two angles (θ_o and θ_e). The first bounds the *normals* of the emitters. For example a cluster containing a single flat emitter (like a quad or triangle) will have an axis equal to the normal and a zero angle. As we group more lights into the cluster, the cone should represent a tight bound on the possible normals. Other light types like spheres, points or meshes typically will start with a normal orientation bound covering the entire sphere.

The second bound we track represents the emission profile of each emitter. For uniform emitters this angle will be $\pi/2$, but it could be smaller for lights with custom profiles or spotlights. In the case of a mesh light, we use the fact that the mesh light is also clustered using a hierarchy and that we can use its root node directly. We summarize examples for various emitter types in Table 1. We note that the particular choice of angles for spotlights reflects that in our renderer spot lights are represented as discs with a restricted emission profile.

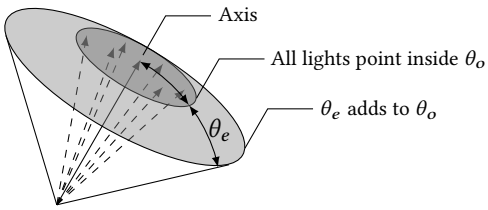

Fig. 2. The emission extent θ_e expands the bounds of the normal orientation bounding cone θ_o. Several lights can be enclosed in this orientation bounding structure.

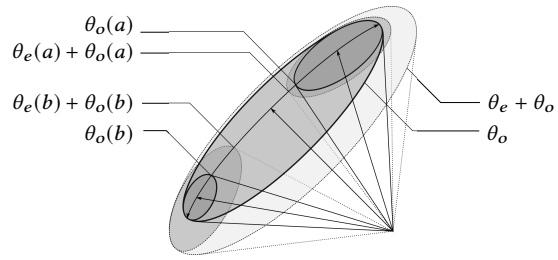

Fig. 3. The axis of the union is chosen in the middle of the arc that covers both $\theta_o(a)$ and $\theta_o(b)$ input cones. From this vector the new angle θ_o is half the covering arc and $\theta_e = \max\{\theta_e(a), \theta_e(b)\}$

Dividing the bounds in these two angles allows us to account for the importance falloff from θ_o to $\theta_o + \theta_e$ following a cosine law. This is important for flat emitters, which are the basic primitives in mesh lights, one of the primary motivations for the work in this paper.

From the bounds of individual clusters, we also need a way to *merge* clusters together. Our procedure for merging the bounding cones is summarized in Algorithm 1. The basic idea is to compute a new axis and θ_o angle to cover all the emitters with the smallest possible cone. The new angle θ_e is chosen as the maximum emission profile of the two clusters. Our method is a faster, greedy alternative to the minimum bounding cone method from [Barequet and Elber 2005]. While this operation is not strictly associative in all cases, we have found it to be stable enough in practice.

Algorithm 1 Compute the union of two non-empty bounding cones a and b as tuples of three values: axis, θ_o and θ_e

```
function ConeUnion(a, b)
    if θo(b) > θo(a) then
        Swap(a, b)
    end if
    θd ← arccos(axis(a) · axis(b))
    θe ← max{θe(a), θe(b)}
    if min{θd + θo(b), π} ≤ θo(a) then
        return {axis(a), θo(a), θe}                    ▷ Bounds a already covers b
    else
        θo ← (θo(a) + θd + θo(b))/2                    ▷ New cone over a and b
        if π ≤ θo then
            return {axis(a), π, θe}
        end if
                                  ▷ Rotate a's axis towards b's axis so θo covers both from it
        θr ← θo − θo(a)
        axis ← Rotate(axis(a), axis(a) × axis(b), θr)
        return {axis, θo, θe}
    end if
end function
```

Figure 3 shows how Algorithm 1 merges two input cones. This is clearly the smallest cone covering the orientation directions and emission profiles of the operands, but if we apply this union incrementally to a collection of cones the result is likely not optimal. Nevertheless, it is good enough for a fast one-pass approach. An improved (but slower) system could compute the new centroid axis in a first pass and then find the minimum angle θ_o covering the entire collection.

4.2 Tree Construction

We organize lights into a binary tree very similar to the BVH typically used for ray intersection. We choose to organize our builder around spatial binning [Wald 2007]. The only difference is that we track the additional information of the orientation bounds and sum the light powers during the construction. In ray tracing, the *Surface Area Heuristic* aims to minimize the cost of intersection (represented by the number of primitives in a cluster). The probability of intersecting a smaller cluster given that we have intersected its parent is represented by the ratio of surface areas which produces a simple formula to predict the relative cost of a split.

For our application, we would like to evenly distribute the emitter power within the tree. The decision about how to partition any given cluster into smaller ones should also take into account that it is preferable to keep lights with similar orientations together (see Figure 5). The overall surface area of a cluster is also important as it figures implicitly in our cluster importance measure that drives the traversal (Section 5). The bigger this area, the more uncertainty about the position of emitters inside the cluster. Intuitively, lights that are spatially close together should be clustered together because the geometric falloff of $1/d^2$ will be similar for all lights in that cluster. Likewise, given similar distances to a group of lights, we want to cluster those oriented similarly as they will make similar contributions.

After construction, each level of the hierarchy stores spatial and orientation bounds, as well as the energy total for all lights contained below. These quantities are used by the cluster importance measure explained in Section 5.

4.3 Orientation Bounds Area Measure

Before describing the SAOH equation used for splitting the tree during top-down construction, we detail how we turn an orientation bounds into a scalar measure that predicts its relative impact on the scene.

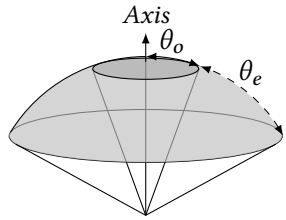

Fig. 4. The dark disc bounded by θ_o contributes its whole solid angle to the measure, while the light sector spawned by θ_e adds its cosine weighted solid angle.

The surface area of a box is a good indicator of the overall solid angle of that region for points anywhere in space. A box with smaller surface area can be expected to have proportionally less impact on a scene. For the orientation bounds we must account both for the cone of normals (θ_o), as well as the possible falloff from that cone (θ_e) which accounts for the emission profile (usually $\pi/2$). Every possible normal within the θ_o cone affects a range of directions. This is important, because as our intuition shows in looking at Figure 4, if θ_o is already small, reducing it to half barely changes the total measure of the bounds, which would be given by:

$$M_\Omega = 2\pi \left[(1 - \cos\theta_o) + \int_{\theta_o}^{\theta_w} \cos(x - \theta_o)\sin(x)dx \right].$$

where $\theta_w = \min(\theta_o + \theta_e, \pi)$. This integral has a simple solution:

$$M_\Omega = 2\pi\,(1 - \cos\theta_o) + \frac{\pi}{2}\Big(2\theta_w \sin\theta_o - \cos\left(\theta_o - 2\theta_w\right) - 2\theta_o \sin\theta_o + \cos\theta_o\Big). \quad (1)$$

For the common case of uniform emitters ($\theta_e = \pi/2$), the measure M_Ω varies smoothly between π (for a flat emitter) to 4π for a complete sphere of directions.

4.4 Surface Area Orientation Heuristic

During the top-down tree construction, we measure the relative influence of a cluster to its parents by comparing the ratio products of the area measure M_A (the area of the bounding box) and M_Ω (detailed in Equation 1). These ratios are weighted against the energy in each cluster. All the axes of the 3D bounding box are explored. At each split candidate on axis i and location s, the cost function

$$C_{\text{split}}(i, s) = K_r(i)\frac{E_L M_A(L) M_\Omega(L) + E_R M_A(R) M_\Omega(R)}{M_A M_\Omega}, \quad (2)$$

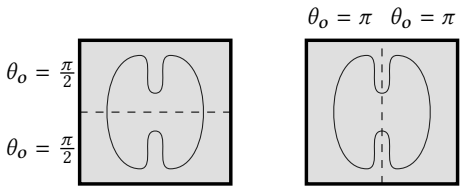

Fig. 5. A cross section of a mesh light inside a cluster. Our SAOH heuristic favors the split on the left since it also divides the orientation space in half. This enables better culling during traversal.

Fig. 6. We show the effect of the SAOH heuristic on sampling. The model is lit by a toroidal mesh light. The difference is subtle but noticeable.

will determine the cost of splitting the current list of emitters into smaller clusters L and R, where E_l and E_r are the corresponding energy sums. We also consider a regularization factor

$$K_r(i) = \frac{\text{length}_{\text{max}}}{\text{length}(i)}$$

for the axis i being considered, where $\text{length}_{\text{max}}$ is the maximum length of the cluster's 3D box. This penalizes the choice of very thin boxes which hurt sample stratification and are not well represented by their bounding cones when seen from a shading point (see section 5.1).

The cost function tries to balance and minimize the probability of sampling each of the two resulting branches given a random shading point. Termination of the tree construction is driven by either reaching a single emitter per leaf, or by finding a point where the splitting cost is never better than the cost of producing a leaf:

$$C_{\text{leaf}} = E_c$$

which is just the total energy of the current cluster. This small change to the SAH lets us better equalize the overall contribution of each cluster to the scene and keep lights with similar orientations in different branches of the tree. While the impact on random layouts of lights is small, it is very helpful in mesh lights where there is an orientation locality on common surfaces, as shown in Figure 6.

5 LIGHT HIERARCHY TRAVERSAL

5.1 Cluster Importance for Surfaces

Given a hierarchy of light clusters, we must have a way to measure at every node the relative contribution that cluster could make to the current shading point (Figure 7). The importance is computed from both the spatial and orientation cluster bounds, and a shading point. Our heuristic considers both geometric decay due to the inverse square distance, and the cosine factor from the orientation bounds of the cluster. Let θ_u be the angle of a cone that would cover the entire bounding box as seen from the shading point, θ_i the incident angle from the shading point to the

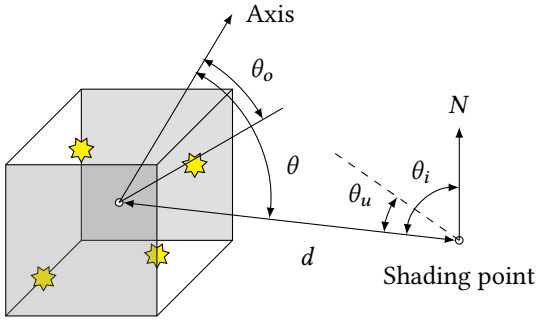

Fig. 7. Using the cluster bounds to compute the importance from a shading point. The angle θ_u captures the solid angle of the entire box, because emitters could be at any location within it. Note that θ_e is only used to clip the angle $\theta - \theta_o - \theta_u$

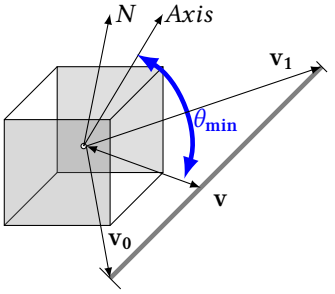

Fig. 8. In participating media, getting a conservative estimate for the orientation term is a maximization problem on the cosine v (running along the line integral) forms with the axis of the orientation bounds.

cluster's center, and d the distance of this segment. We define our measure as

$$I_s = \frac{f_a |\cos\theta_i'| \, E}{d^2} \times \begin{cases} \cos(\theta') & \theta' < \theta_e \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where E is the total energy of the emitters inside the cluster, $\theta_i' = \max\{\theta_i - \theta_u, 0\}$ is the minimum incidence angle and $\theta' = \max\{\theta - \theta_o - \theta_u, 0\}$ is the minimum angle an emitter's normal would form with the direction to the shading point as shown in Figure 7. That is, the angle of the cluster orientation cone (Axis, θ_o) with the vector from the center towards the shading point, minus θ_u. Finally the term $f_a|\cos\theta_i'|$ is a conservative and arbitrary approximation of the surface BSDF f_a times the irradiance. We have found that a diffuse approximation here is nicely complemented by BRDF sampling via MIS, but taking into account an approximate form of the specular response is an interesting avenue for future work.

The location of an emitter inside the box is unknown so we lack certainty about the angle formed between its direction to the shading point and the box's orientation axis. We use the center of the box and consider the θ_u cone uncertainty angle. The orientation part of the measure needs the angle from the orientation cone $\theta - \theta_o$, but we need to also subtract θ_u to make a conservative guess. This results in the minimum angle any emitter in the cluster will form with the direction towards the shading point. A similar precaution is taken for the irradiance angle θ_i'.

Finally, we compute the maximum cosine for any emitter. The emission profile angle θ_e is used for clamping as shown in Equation 3. When smaller than $\pi/2$, indicative of one or more spot lights, the cosine factor might not be an exact representation of the emitted light, but we found it to be safe in practice.

We want emphasize that the energy term E used in Equation 3, when computed for a single emitter, is not the total energy emitted in all directions. Instead, we use the maximum radiance emitted in some direction integrated over the emitting area. This is consistent with our equations for importance, which try to approximate the energy arriving at the shading point.

5.2 Cluster Importance for Participating Media

When computing lighting in participating media, we no longer have a single shading point but a line. We assume that the direct lighting will be computed through a form of equiangular sampling [Kulla

and Fajardo 2012]. In this case, the inverse squared falloff actually becomes closer to inverse linear. This is due to the following integral over the line:

$$\int_a^b \frac{1}{x^2 + d^2} dx = \frac{\theta_b - \theta_a}{d} \tag{4}$$

where d is the shortest distance between a point and the current ray. This normalization term is what remains after equiangular integration for a specific light. We therefore adjust our importance heuristic to try and account for a $1/d$ falloff instead of a $1/d^2$ falloff.

In Figure 8 we show how the vector $\mathbf{v}$ runs along the volume line integral and on a plane with normal N containing the line and the cluster center. The point along the ray that minimizes the distance to the cluster (d_{min}) might not coincide with the minimum angle (θ_{min}). We compute both points independently to get a conservative importance measure. That is, the geometric factor is computed with the minimum distance and the orientation factor with the minimum angle, even though they might not correspond to the same location.

The orientation factor given the bounds axis $\mathbf{a}$ comes from the maximum dot product $\mathbf{v} \cdot \mathbf{a}$, where $\mathbf{v} = \mathbf{o_0} \cos\varphi + \mathbf{o_1} \sin\varphi$ is a parametric vector over an ortho-normal basis $\mathbf{o_0}, \mathbf{o_1}$ coming from $\mathbf{v_0}, \mathbf{v_1}$. This basis can be obtained from $\mathbf{v_0}/\|\mathbf{v_0}\|$ and the ortho-normalized vector from $\mathbf{v_1}$. From this point we will assume $\mathbf{v_0}$ and $\mathbf{v_1}$ are normalized. Applying derivative cancellation of $\mathbf{v} \cdot \mathbf{a}$ for some angle φ_0, if the maximum cosine at the boundary is $b_{\max} = \max\{\mathbf{v_0} \cdot \mathbf{a}, \mathbf{v_1} \cdot \mathbf{a}\}$, then we can compute θ_{min}:

$$\cos\varphi_0 = \frac{\mathbf{o_0} \cdot \mathbf{a}}{\sqrt{(\mathbf{o_0} \cdot \mathbf{a})^2 + (\mathbf{o_1} \cdot \mathbf{a})^2}} \tag{5}$$

$$\cos\theta_{min} = \begin{cases} b_{\max} & \mathbf{o_1} \cdot \mathbf{a} < 0 \vee \mathbf{v_0} \cdot \mathbf{v_1} > \cos\varphi_0 \\ (\cos\varphi_0\ \mathbf{o_0} + \sin\varphi_0\ \mathbf{o_1}) \cdot \mathbf{a} & \text{otherwise.} \end{cases} \tag{6}$$

When $\mathbf{o_1} \cdot \mathbf{a} < 0$ the derivative cancels at a minimum and has to be ignored. Then, the importance measure for the cluster from a volumetric segment is just:

$$I_v = \frac{\cos\left(\max\{\theta_{min} - \theta_o - \theta_u, 0\}\right)\ E}{d_{min}}. \tag{7}$$

Note, we clip the result to 0 if $\theta_{min} - \theta_o - \theta_u$ exceeds θ_e as in Equation 3, which we omit here for brevity.

5.3 Importance Sampled Tree Traversal

We traverse our binary tree stochastically to randomly choose a single light. This can be viewed as a 1D version *hierarchical sample warping* [Clarberg et al. 2005]. At every branch in the tree we compute an importance term I by either the surface or volume formulation described earlier to approximate the potential contribution of lights within the cluster. We choose between the left or right subtrees by assigning the probability $P_{\mathrm{L}} = I_{\mathrm{L}}/(I_{\mathrm{L}} + I_{\mathrm{R}})$ to the left child and similarly P_{R} to the right. A single random number ξ that we rescale after each decision guides the traversal until we reach a leaf. At the leaves of the tree, if we have more than one light to choose from we simply build a discrete probability distribution on the fly, again using our importance measure. The pseudo-code in Algorithm 2 summarizes this process. Obtaining the corresponding probability for the chosen light is equally simple.

Note that the importance measure logic is the same regardless of being applied to a cluster or an emitter, which is a special case of a cluster with just one member. The better I approximates the actual contribution of lights in the cluster, the more variance reduction we can expect. Our measure from the previous section works well if the cluster is far away from the shading point. When the

Algorithm 2 Choose a random light from a tree T for a given shading point P and a random number ξ.

```
procedure PickLight(T, P, ξ)
    if IsLeaf(T) then
        PDF ← [ ]
        for e in Emitters(T) do
            PDF ← PDF + [Importance(e)]
        end for
        CDF ← Cumulative(Normalize(PDF))
        return SampleCDF(CDF, Emitters(T), ξ)
    else
        I_L ← Importance(Left(T))
        I_R ← Importance(Right(T))
        if ξ < I_L/(I_L + I_R) then
            ξ ← ξ(I_L + I_R)/I_L
            return PickLight(Left(T), P, ξ)
        else
            ξ ← (ξ(I_L + I_R) − I_L)/I_R
            return PickLight(Right(T), P, ξ)
        end if
    end if
end procedure
```

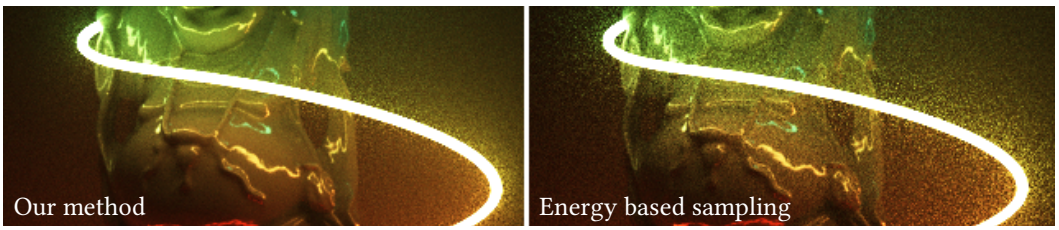

Fig. 9. A long spiral mesh light and participating media. We compare against energy based sampling. Both images rendered in 2 minutes.

cluster is close or when the bounds contain the shading point, the measure is less effective. The splitting heuristic described in the following section can mitigate this weakness.

On its own, this stochastic traversal allows unbiased rendering of scenes with many lights. It works particularly well for collections of emitting triangle lights (mesh lights) as seen in Figure 9.

5.4 Adaptive Splitting Heuristic

One weakness of the purely stochastic tree traversal is that the predicted importance I is sometimes inaccurate, particularly near the root of the tree where the bounds are large. This can lead some samples to be directed towards subtrees that will ultimately make smaller contributions. We mitigate this weakness by forcing *splitting* during traversal until we gain confidence over the quality of our importance criteria I. This process resembles the *cut* selection from the Lightcuts algorithm, except that instead of computing final lighting from the chosen clusters, we continue with stochastic traversal to sample lights from the actual emitters in the leaves. This means our cuts can be quite

shallow without affecting the accuracy of the result. They only serve as a way to force exploration to multiple branches of the tree at once.

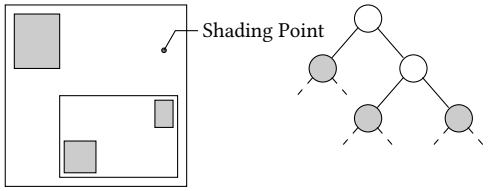

Fig. 10. We draw three samples from the gray nodes, the other two upper in the tree are split according to our heuristic. From the depicted shading point the estimated variance scores too high to reliably work with a single random light from the whole tree.

The decision to split traversal into both branches is based on the estimated variance of the lighting within it. We consider two sources of variance: the energy differences between the emitters

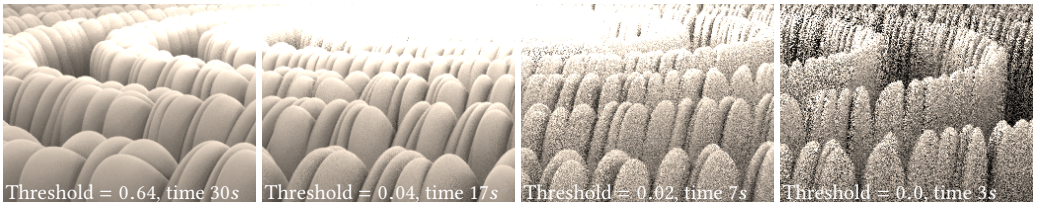

Fig. 11. A spiral of rounded diffuse objects, each with a point light located above for a total of 10k lights. Splitting thresholds in decreasing order show the response in render time and quality. This shows how using more than one random light per point can be very beneficial.

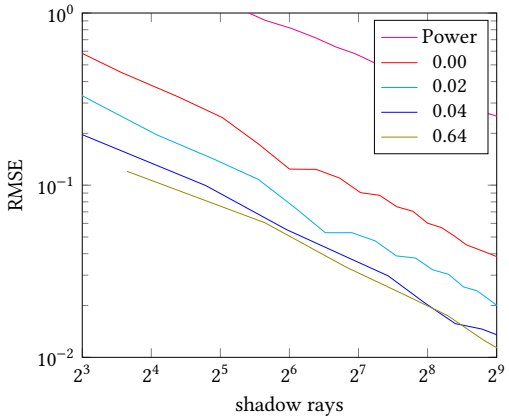

Fig. 12. RMS Error as a function of shadow rays for the image from Figure 11. Naive power based sampling has high error, even compared to ours without any splitting (0.0). Increasing the split factor improves the error convergence until we get to diminishing returns.

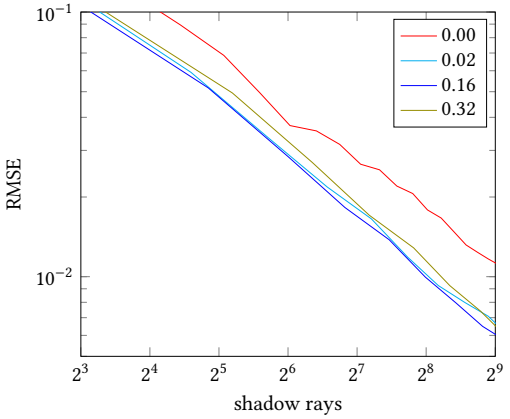

Fig. 13. RMS Error as a function of shadow rays for the image from Figure 14. Here, the ideal split rate is around 0.16; additional splitting reduces the error at too great a cost to be worthwhile.

in the tree and the geometric term $1/d^2$. The variance in emitted power is stored explicitly in each cluster. For the geometric term, we can compute the continuous mean and variance of the geometric term g as follows:

$$E[g] = \frac{1}{b-a}\int_a^b 1/x^2 dx = \frac{1}{ab}, \tag{8}$$

$$\begin{aligned} V[g] &= E[g^2] - E[g]^2 \\ &= \frac{1}{b-a}\int_a^b 1/x^4 dx - \frac{1}{a^2b^2} \\ &= \frac{b^3-a^3}{3(b-a)a^3b^3} - \frac{1}{a^2b^2}, \end{aligned} \tag{9}$$

where (a, b) is the range where the distance to an emitter in the cluster varies. This can be simply obtained from the cluster center and the radius of the bounding sphere. The total light variance from the cluster is just the product variance between $V[g]$ and $V[e]$, the emitter energy variance recorded in the node:

$$\sigma^2 = \left(V[e]V[g] + V[e]E[g]^2 + E[e]^2V[g]\right), \tag{10}$$

where $V[e]$ is the precomputed variance of the emitters energy stored in the cluster. We remap this value to the $[0, 1]$ range using $\sqrt[4]{1/(1+\sigma)}$ so the user can control the amount of splitting in a simple way. At any point in the traversal where this estimate is under the threshold we continue traversing on *both* subtrees. Otherwise we use stochastic traversal as seen earlier and disable further evaluation of the split heuristic.

Algorithm 3 Light traversal with splitting. Returns a list of lights to use from the tree T for the shading point P.

```
procedure GetLights(T, P, ξ)
    if SplitMeasure(T) < UserThreshold then
        L ← GetLights(Left(T), P, ξ) ∪ GetLights(Right(T), P, ξ)
    else
        L ← { PickLight(T, P) }                          ▷ One light is enough
    end if
    return L
end procedure
```

For the case of participating media, we have found that although our importance measure switches to inverse linear decay, we saw better performance from using the split heuristic above (which assumes a quadratic decay). In both cases, we are ignoring the effect of transmittance which tends to further dampen the falloff. In practice we have found that using the same formula as for surfaces prevents over-splitting, though it remains to be seen if an even better heuristic is possible.

The effects of this adaptive heuristic can be seen in Figure 14. In an equal time comparison, we have found that splitting always outperforms purely stochastic traversal.

6 IMPLEMENTATION DETAILS

We have integrated this algorithm into a production path tracer designed for film production. We support the use of multiple lights per shading point, and therefore the adaptive splitting works very well in this case. In the case of mesh lights, our system assumes that it can draw individual

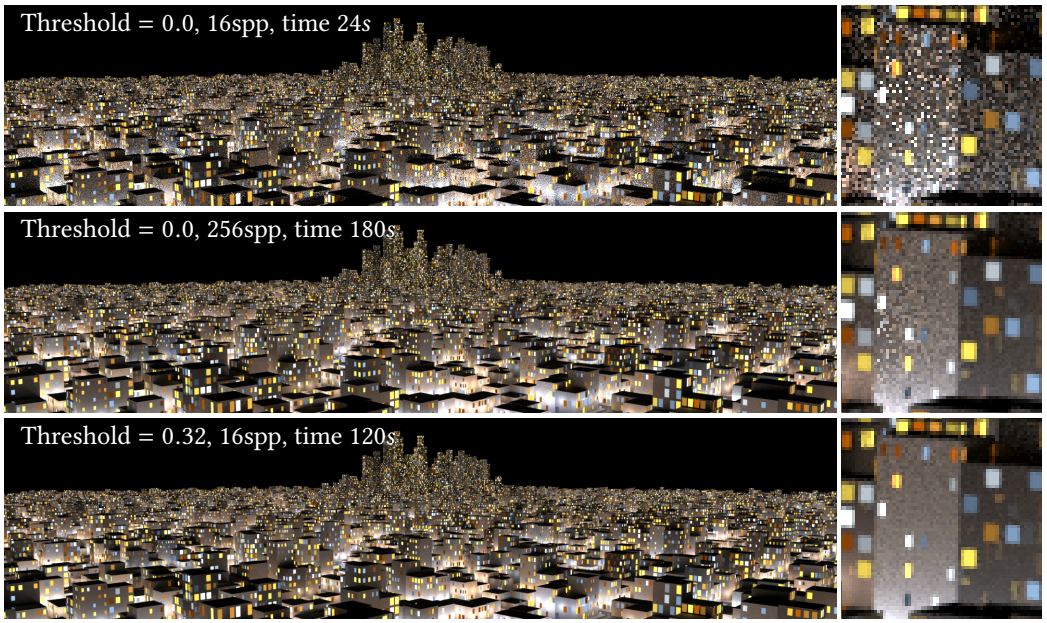

Fig. 14. A procedural city with more than 300k lights and surface rendering only. The top two rows were rendered without splitting and with one shadow ray per shading point. The bottom row was rendered with splitting which clearly illustrates its advantage over simply increasing pixel sampling rates.

samples and therefore use of the adaptive split heuristic is disabled. Despite this, our system offers much better performance compared to naive approaches such as uniform sampling.

When splitting is disabled, our cluster importance measure may not be fully robust for nearby shading points. The center of the cluster is not representative of the emitter positions over short distances. We work around this by clamping the distance to half the radius of the cluster. This prevents the geometric term from increasing uncontrollably when it is not reliable.

Computing the PDF in Algorithm 2 is trivial, and splitting does not alter it since it is a deterministic process. However, to use our method with MIS, we need to be able to evaluate the probability of an arbitrary light source. This requires the ability to reconstruct the path towards a specific leaf. We chose to use bit-trails [Laine 2010] to encode the left/right decisions required to reach each light source as an integer. This provides a compact encoding that does not require the presence of parent pointers in the light tree. When a ray associated with a BSDF sample intersects a light source, we use the bit-trail path through the tree to traverse our light hierarchy while computing the probability, similarly to Algorithm 2.

7 RESULTS

This system has been in use at our studio for over a year, successfully empowering artists to freely add as many lights to a scene as they want. Since it can be used for both mesh lights and large light collections, the benefit has been two-fold. Mesh lights do not use the splitting mechanism, since only one sample at a time is expected in our API, but they are the biggest beneficiaries from the orientation term of our importance measure. In Figure 15 we show the impact of the orientation term on top of the geometric one for a mesh light.

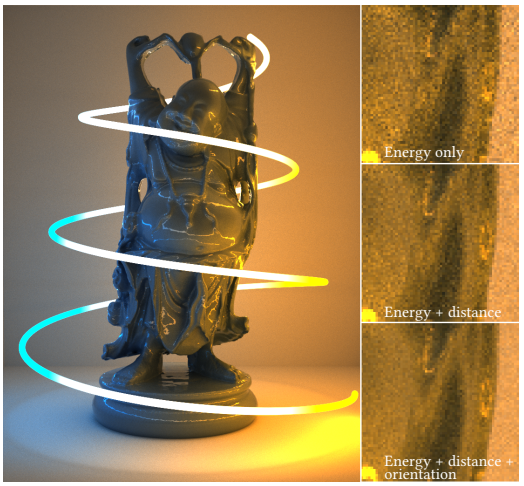

Fig. 15. Our algorithm applied to a mesh light. The three insets on the right show the effect of three different components of our importance measure from Equation 3.

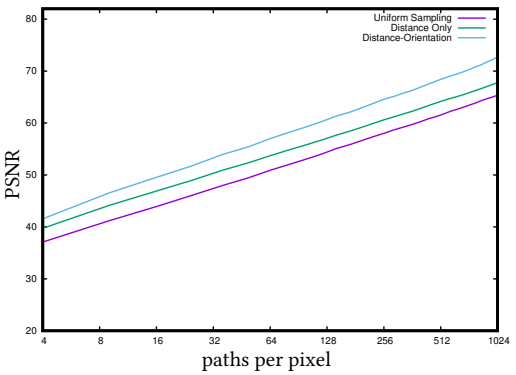

Fig. 16. Convergence curves for Figure 15 with direct illumination only. Showing Peak Signal Noise Ratio. Our sampling improves the image by 5 dB with distance sampling and another 3 dB from orientation.

In the case of large light collections like shown in Figure 14, splitting and its adaptive nature has proven very useful to avoid having to increase camera rays to compensate for the extra noise. In fact, when combined with variance driven adaptive sampling, it allows renders to finish *faster* than if splitting were disabled. Nearby lights are always sampled with probability 1.0 which greatly reduces variance. Figure 18 shows how the number of shaded lights increases as the shading point gets closer to an area very dense in lights. This is important because not all the points on the scene need to trace the same amount of shadow rays. At the same time, over-splitting can be counterproductive as well, as shown in Figure 13. A threshold of 1.0 would cause all lights to be shaded, which will produce very low error, but at a high cost in number of rays. We can see this effect in Figure 12 — the error decreases as we raise the split threshold until the point where we start firing so many rays that the amount of error per traced ray causes the curves to go back up again.

Intuition suggests that jumps in the number of shaded lights could lead to discontinuities, but in practice the transitions produced by the variance based heuristic (Section 5.4) are smooth enough to render these transitions invisible. In practice we are further assisted by adaptive pixel sampling which equalizes the variance across the image. We only used fixed numbers of camera samples for the results presented in this paper.

We highlight another seemingly simple example in Figure 17 where a million lights are added to the ceiling of a Cornell Box. In this case, our algorithm is able to both render the far-field effect of the many lights by choosing only a handful when shading the floor, as well as the near field behavior very close to the light sources where it selects a few dozen of the nearest lights by tree splitting.

Finally, we show a comparison of our system to the spatial light distribution implemented in PBRT [Pharr et al. 2016] in Figure 19. The algorithm used by PBRT is based on a 64^3 grid where each voxel builds a PDF over all the lights using the existing light sample methods to estimate their potential contribution to random points in that voxel. Their method is quite elegant in that it requires very little specific knowledge about the lights, but when the light collection becomes

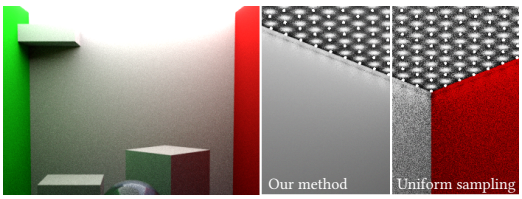

Fig. 17. A cornell box lit by a million lights. This is a case where lights are non-local. The under-exposed detail on the right half shows the layout.

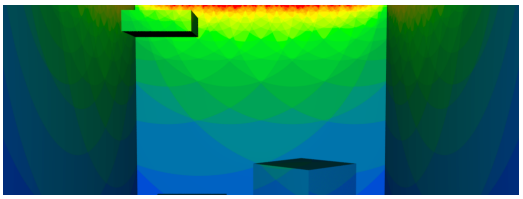

Fig. 18. In this heat-map, color indicates the light count per shading point on the left scene: the red areas in the ceiling represent around 28 lights; green around 10; and dark blue 2-3 (0.85 split threshold).

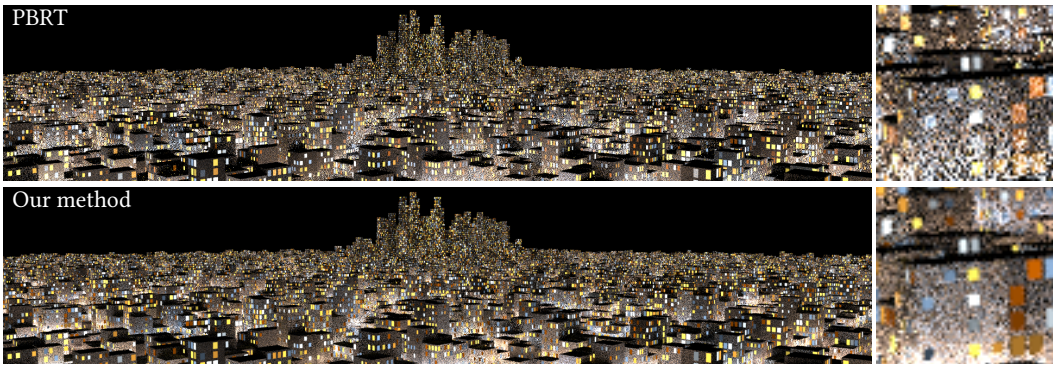

Fig. 19. A comparison between PBRT on the top row and our method below. Both render with one shadow ray per shading point (which means splitting is disabled for ours) and 16 samples per pixel. The cropped regions on the right illustrate the improved image quality from our technique. The render times are 90 minutes for PBRT (due to its lengthy initialization) and 22 seconds for our renderer.

large, the initialization time is prohibitively long. In contrast, our tree construction is quite fast and only depends on the number of emitters. Once PBRT populates its grid, it is able to render faster since it just re-uses those precomputed PDFs. Nonetheless, at equal sample counts, even ignoring the initialization time our algorithm clearly shows an improvement.

While we have described our system as using binary trees, our practical implementation takes advantage of SIMD instructions and builds a four-wide tree. This change reduces the overall depth of the tree, which improves stratification by reducing the amount of sample stretching in Algorithm 3. It does mean that the splitting operation follows four subtrees instead of two, but this can be compensated by adjusting the split threshold.

8 FUTURE WORK

The basic framework we have presented here presents numerous avenues for future work. In the context of acceleration structures for ray tracing, several researchers have observed that bottom-up construction can lead to higher quality trees [Gu et al. 2013; Walter et al. 2008]. The same algorithms should transfer fairly naturally to our case as well. The cost function (Equation 2) has worked well, but further exploring the relationship between this cost metric and the importance metric could lead to even better performance.

In highly occluded environments, the visibility function can be a major source of noise which is not addressed by our work. Approaches that try to incorporate visibility progressively [Donikian

et al. 2006; Vévoda et al. 2018] could in theory be combined with our work and be included in the cluster importance. The concurrent work by Vévoda et al. [2018] is particularly interesting though we believe their requirement on spatially caching the visibility information presents a challenge in scenes with large spatial extent or volume effects that fill the scene. Finding a spatial caching structure that balances resolution against memory usage in an intuitive way is still a challenge.

The diffuse approximation we make in our cluster importance could be improved by trying to estimate a cheap approximation of the actual BRDF. Recent representations suitable for real time lighting [Dupuy et al. 2017; Heitz et al. 2016] could be beneficial to our work as well.

Perhaps more importantly, finding the optimal value for the split threshold is still a manual process we would like to automate in a principled way. While in our implementation the default of 0.85 has proven to be a conservative starting point, optimally tuning this number can be difficult for artists who usually do not have time to devote to experimenting with rendering parameters. Even though we believe that splitting is a key benefit to our algorithm, finding a parameter-free policy would be desirable.

ACKNOWLEDGMENTS

We want to thank Matt Pharr not only for thorough revisions of this document and his useful suggestions, but also for helping us to export the procedural city in Figure 19 to PBRT for comparison. We also thank Clifford Stein for his careful final revision.

REFERENCES

Gill Barequet and Gershon Elber. 2005. Optimal bounding cones of vectors in three dimensions. *Inf. Process. Lett.* 93, 2 (2005), 83–89. DOI: http://dx.doi.org/10.1016/j.ipl.2004.09.019

Jacco Bikker. 2007. Real-time Ray Tracing Through the Eyes of a Game Developer. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing (RT '07)*. IEEE Computer Society, Washington, DC, USA, 1–10. DOI: http://dx.doi.org/10.1109/RT.2007.4342584

Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. 2005. Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 24, 3 (2005), 1166–1175.

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.

Michael Donikian, Bruce Walter, Kavita Bala, Sebastian Fernandez, and Donald P. Greenberg. 2006. Accurate Direct Illumination Using Iterative Adaptive Sampling. *IEEE Transactions on Visualization and Computer Graphics* 12, 3 (May 2006), 353–364. DOI: http://dx.doi.org/10.1109/TVCG.2006.41

Jonathan Dupuy, Eric Heitz, and Laurent Belcour. 2017. A Spherical Cap Preserving Parameterization for Spherical Distributions. *ACM Trans. Graph.* 36, 4, Article 139 (July 2017), 12 pages. DOI: http://dx.doi.org/10.1145/3072959.3073694

Petra Gospodnetić. 2017. GSoC 2017 with AppleseedHQ. (2017). https://medium.com/@petragospodneti/gsoc-2017-with-appleseedhq-cc5f33d04170

Yan Gu, Yong He, Kayvon Fatahalian, and Guy Blelloch. 2013. Efficient BVH Construction via Approximate Agglomerative Clustering. In *Proceedings of the 5th High-Performance Graphics Conference (HPG '13)*. ACM, New York, NY, USA, 81–88. DOI: http://dx.doi.org/10.1145/2492045.2492054

Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. 2016. Real-time Polygonal-light Shading with Linearly Transformed Cosines. *ACM Trans. Graph.* 35, 4, Article 41 (July 2016), 8 pages. DOI: http://dx.doi.org/10.1145/2897824.2925895

Alexander Keller, Carsten Wächter, Matthias Raab, Daniel Seibert, Dietger van Antwerpen, Johann Korndörfer, and Lutz Kettner. 2017. The Iray Light Transport Simulation and Rendering System. *CoRR* abs/1705.01263 (2017). http://arxiv.org/abs/1705.01263

Christopher Kulla and Marcos Fajardo. 2012. Importance Sampling Techniques for Path Tracing in Participating Media. *Comput. Graph. Forum* 31, 4 (June 2012), 1519–1528. DOI: http://dx.doi.org/10.1111/j.1467-8659.2012.03148.x

Samuli Laine. 2010. Restart Trail for Stackless BVH Traversal. In *Proceedings of the Conference on High Performance Graphics (HPG '10)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 107–111. http://dl.acm.org/citation.cfm?id=1921479.1921496

Eric Paquette, Pierre Poulin, and George Drettakis. 1998. A Light Hierarchy for Fast Rendering of Scenes with Many Lights. *Computer Graphics Forum* (1998). DOI: http://dx.doi.org/10.1111/1467-8659.00254

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

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Trans. Graph.* 15, 1 (Jan. 1996), 1–36. DOI: http://dx.doi.org/10.1145/226150.226151

Yusuke Tokuyoshi and Takahiro Harada. 2016. Stochastic Light Culling. *Journal of Computer Graphics Techniques (JCGT)* 5, 1 (29 March 2016), 35–60. http://jcgt.org/published/0005/01/02/

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 (SIGGRAPH '95)*. ACM, New York, NY, USA, 419–428. DOI: http://dx.doi.org/10.1145/218380.218498

Nathan Vegdahl. 2014. Efficient sampling of many lights. (2014). http://ompf2.com/viewtopic.php?t=1938

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4 (2018).

Petr Vévoda and Jaroslav Křivánek. 2016. Adaptive Direct Illumination Sampling. In *SIGGRAPH ASIA 2016 Posters (SA '16)*. ACM, New York, NY, USA, Article 43, 2 pages. DOI: http://dx.doi.org/10.1145/3005274.3005283

Ingo Wald. 2007. On Fast Construction of SAH-based Bounding Volume Hierarchies. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing (RT '07)*.

Bruce Walter, Kavita Bala, Milind Kulkarni, and Keshav Pingali. 2008. Fast Agglomerative Clustering for Rendering. (09 2008), 81 – 86.

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 (July 2005), 1098–1107. DOI: http://dx.doi.org/10.1145/1073204.1073318

Gregory J. Ward. 1994. *Adaptive Shadow Testing for Ray Tracing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 11–20. DOI: http://dx.doi.org/10.1007/978-3-642-57963-9_2