High-Performance Graphics (2019)
T. Foley and M. Steinberger (Editors)

Stochastic Lightcuts

Cem Yuksel

University of Utah, UT, USA

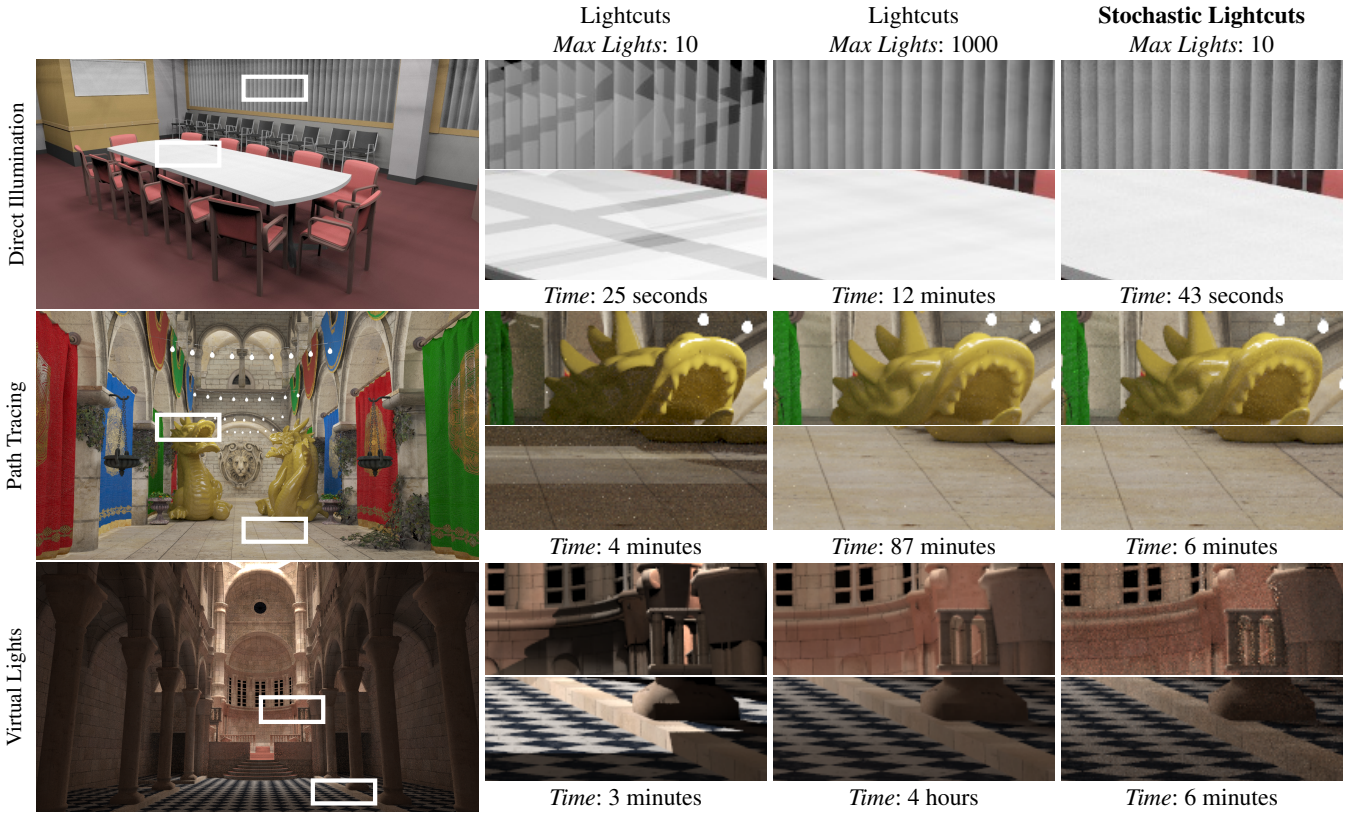

Figure 1: *Comparison of lightcuts and our stochastic lightcuts for direct illumination estimation with different rendering methods: (top) direct illumination only from 1400 light sources, (middle) path tracing up to 5 bounces with 1644 light sources, and (bottom) one million virtual lights, using 64 samples per pixel. Notice that limiting lightcuts to 10 light samples produces a substantial amount of error and correlation artifacts. Using up to 1000 light samples reduces the error, but still leads to visible flickering and substantially increases the render time. Our stochastic lightcuts method can produce a fast, temporally-stable, and low-noise lighting estimation with only 10 samples.*

Abstract

We introduce stochastic lightcuts by combining the lighting approximation of lightcuts with stochastic sampling for efficiently rendering scenes with a large number of light sources. Our stochastic lightcuts method entirely eliminates the sampling correlation of lightcuts and replaces it with noise. To minimize this noise, we present a robust hierarchical sampling strategy, combining the benefits of importance sampling, adaptive sampling, and stratified sampling. Our approach also provides temporally stable results and lifts any restrictions on the light types that can be approximated with lightcuts. We present examples of using stochastic lightcuts with path tracing as well as indirect illumination with virtual lights, achieving more than an order of magnitude faster render times than lightcuts by effectively approximating direct illumination using a small number of light samples, in addition to providing temporal stability. Our comparisons to other stochastic sampling techniques demonstrate that we provide superior sampling quality that matches and improves the excellent convergence rates of the lightcuts approach.

1. Introduction

The problem of rendering with a large number of light sources (a.k.a. the many-lights problem) is often considered in the context of global illumination computation with many virtual light sources [DKH*14]. Yet, the many-lights problem has growing applicability in computer graphics, as we continue to render more complex scenes with more actual light sources in them.

Lightcuts [WFA*05] is one of the first methods introduced for efficiently handling many lights, and it is still a preferred method. On the other hand, lightcuts, like most other scalable lighting solutions, is temporally unstable due to sampling correlation, which leads to flickering and hinders its use in practice.

In this paper, we introduce *stochastic lightcuts* that incorporates stochastic sampling into the illumination estimation framework of lightcuts to eliminate the sampling correlation. To minimize the sampling noise, we introduce a robust hierarchical sampling strategy that combines the benefits of importance sampling, adaptive sampling (provided by lightcuts), and stratified sampling (using a light tree). Our method only modifies the light sampling order of lightcuts, so it does not hinder its flexibility, applicability, or its impressive convergence rate, and it can be easily incorporated into an existing implementation of lightcuts. On the contrary, our stochastic sampling solution introduces extra flexibility and allows using complex light sources that can be difficult to handle using lightcuts. Furthermore, stochastic lightcuts allows placing a user-defined small upper bound on the number of light evaluations per shading computation, which can significantly improve the rendering performance (Figure 1). With these properties, the stochastic lightcuts approach offers the most efficient scalable lighting solution and without the stability problems of existing alternatives.

2. Background

Efficient solutions to the many-lights problem using *lightcuts* [WFA*05] or *matrix row-column sampling* [HPB07] suffer from sampling correlation that lead to temporal instabilities. The *lighting grid hierarchy* method [YY17] provides a temporally-stable solution at the cost of sampling more lights, the cost of which can be amortized by precomputing shadows or approximating them with a few samples [LY19]. The recent *adaptive tree splitting* method provides a hierarchical sampling approach, similar to our stochastic lightcuts solution, but cannot provide the adaptivity of lightcuts and its convergence rate is hindered by its importance formulation.

Our stochastic lightcuts solution extends the lightcuts method [WFA*05]. The lightcuts method starts with building a binary tree, containing the actual light sources at its leaf nodes. Each internal node is used for approximating the illumination from all light sources within its subtree. This is typically implemented as picking one of the lights within the subtree as a *representative light*. The light tree is constructed once, prior to rendering. During rendering, the light tree is evaluated at each shading point, starting from the root node and its representative light. After evaluating the representative light of a node, a conservative error bound is assigned to the node using its bounding box, indicating the maximum possible intensity contribution due to the illumination that can come from the subtree, assuming full visibility (i.e. no shadows). If this error bound is below a user-specified percentage (typically 2%) of the approximated total shading value, the lighting evaluation of the node is accepted. Otherwise, its child nodes are evaluated. Due to the construction of the light tree, one of the child nodes shares the same representative light as the parent node. Therefore, light from that child node can be quickly evaluated without the need for recomputing the shadows of the shared representative light.

A common limitation of all these scalable many-lights solutions is that they require a relatively large number of samples for producing stable/low-noise results. Therefore, simple importance sampling [SWZ96] is still commonplace in practice. Our stochastic lightcuts method, in comparison, can produce temporally stable and relatively low noise results with fewer samples. Therefore, it is highly suitable for rendering algorithms that already rely on multi-sampling, such as path tracing, and significantly improve their performance by providing a low-cost estimation of lighting.

3. Stochastic Lightcuts

We introduce stochastic sampling into the lighting evaluation of lightcuts to eliminate its sampling correlation (Section 3.1), replacing the temporal instabilities of lightcuts with noise. For reducing this noise, we introduce a robust hierarchical importance sampling method (Section 3.2). Stochastic lightcuts can use the same light tree as lightcuts with only minor modifications to the information stored in each node and it can be easily implemented on top of an existing lightcuts implementation (Section 3.3). Therefore, we only describe the differences in the lighting approximation introduced by stochastic light evaluations, as compared to lightcuts.

3.1. Stochastic Sampling with Lightcuts

The sampling correlation of the lightcuts approach is related to the fact that the same light tree is used for rendering the entire image. This is unavoidable in general, because the light tree construction and storage can be expensive. The light tree, containing a representative light per node, forms a spatially varying ordering of light sources. The lighting estimation always begins with the representative light of the root node. Therefore, this light source is always included in the lighting estimation of the entire scene. Similarly, the representative lights at the higher levels of the hierarchy are more likely to be used for the lighting estimation. This predetermined order prior to rendering introduces sampling correlation. By using a conservative error bound, the error within a subtree can be limited to a user-defined percentage of the evaluated pixel color (usually set to 2%). However, this does not bound the total error of the entire lighting estimation. Therefore, errors due to sampling correlation can be substantial and often lead to temporal instability.

This important limitation of lightcuts was also recognized in prior work. The solution that is proposed as a part of the multidimensional lightcuts method [WABG06] was simply storing a list of representative lights per node (such as 32) and randomly selecting one at render time, using the intensities of these pre-selected lights for importance sampling. This approach reduces the correlation, but does not eliminate it. It also inflates the light tree storage. More importantly, due to the performance advantages of sharing representative lights between a node an its parent, this pre-selection is

not performed independently for each node and the sampling correlation is not completely eliminated.

We eliminate the correlation by simply ignoring the representative lights during lighting estimation. Instead, we randomly pick a light source within a given subtree. Any importance sampling scheme can be used here for determining the probability p_i of picking light source i within a subtree. Let p_s be the cumulative probability of all lights within the subtree s. The estimated illumination of the light subtree can be computed by simply scaling the illumination of the light with p_s/p_i. When the error of a node is above the user-defined threshold and we need to evaluate the child nodes, we only need to perform the lighting computation for one of the child nodes, since the other one must contain the light source that was randomly selected for evaluating the parent node. Therefore, just like the original lightcuts method, the light evaluation (including shadow computation) for the parent node is not wasted as we move deeper into the light tree.

This simple modification provides three important benefits:

1. It replaces the predefined order of lights with a randomized order and thereby completely eliminates the sampling correlation.
2. It allows using any type of light source. Since we do not rely on representative lights and we use the actual lights for computation, we impose no restrictions on the light type. The resulting illumination contribution of each light is merely scaled by the corresponding probabilities, as explained above.
3. We can limit the number of lights evaluated during lighting approximation without risking excessive correlation artifacts.

When using representative lights, we must rely on the error bound and traverse as deep into the light tree as necessary, and terminating "cut" selection prematurely (when a maximum number of light samples are computed) can have catastrophic results. This is because the predefined order of representative lights at the higher levels of the light tree can be pathological for estimating the lighting at some points in the scene, and lead to excessive amounts of correlation artifacts. For example, if only a single light evaluation is permitted, the same representative light would be used for the entire scene. Using randomly selected lights, however, we would simply introduce noise by limiting the number of evaluated lights. This can be a significant advantage for some rendering algorithms, such as path tracing, that would ultimately reduce the noise using multi-sampling. Therefore, using path tracing, a faster and noisy lighting estimation would typically be preferable over a slower and more accurate one, and the performance improvements of using fewer light evaluations per lighting estimation can be substantial with minimal or no loss in the final image quality.

3.2. Hierarchical Importance Sampling

The optimal probabilities p_i that would maximize the convergence rate depend on the spatial relationship between the lights and the point where lighting is evaluated. Therefore, the values p_i should ideally vary for each point in the scene. Instead of computing p_i for each light at each shading point, We use a hierarchical sampling strategy. Instead of directly selecting a random light within a subtree, we traverse the subtree step-by-step until we reach a leaf node. Starting with the root node of the subtree, at each step we randomly pick one of the child nodes using importance sampling. The probabilities of picking either child node are assigned based on the expected cumulative illumination contribution of each child node. Note that how these probabilities are computed has utmost importance. Using a poor strategy can result in a worse convergence behavior than the simplest importance sampling scheme that randomly selects a light purely based on its intensity [SWZ96].

3.2.1. Computing Child Node Probabilities

Let p_1 and p_2 represent the probabilities of selecting either one of the child nodes for estimating the lighting at a scene point $\mathbf{x}$. Following the error estimation mechanism of lightcuts, a naive choice for determining the importance weights w_1 and w_2 for picking the child nodes would be using the maximum possible illumination that can come from each child node. The corresponding probabilities are defined as $p_1 = w_1/(w_1+w_2)$ and $p_2 = w_2/(w_1+w_2)$. These weights can be calculated for each node j using a conservative estimate for the maximum geometry term $G_j(\mathbf{x})$, computed using the node's bounding box and its light direction boundsb [WFA*05, EK18]; another conservative estimate for bounding the BRDF (Bidirectional Reflectance Distribution Function) of the material $M_j(\mathbf{x},\omega)$ [WFA*05], where ω is the view direction; the total intensity of the lights within the node $\mathbf{I}_j$; and the minimum distance to the node's bounding box $d_j^{\min}(\mathbf{x})$. Thus, we can write

$$w_j = \frac{F_j(\mathbf{x},\omega)\,\|\mathbf{I}_j\|}{\left(d_j^{\min}(\mathbf{x})\right)^2}\,, \tag{1}$$

where $F_j(\mathbf{x},\omega) = G_j(\mathbf{x})M_j(\mathbf{x},\omega)$ is the reflectance bound, combining the geometry and material terms. Note that out notation slightly differs from the notation of the original lightcuts method [WFA*05], as the inverse-square attenuation factor $1/(d_j^{\min}(x))^2$ is not hidden in $G_j(\mathbf{x})$. Also note that in this notation we do not consider directional light sources, which would not have the inverse-square attenuation factor, so they need to be treated differently.

While the formulation in Equation 1 might appear reasonable at first glance, it would lead to unacceptable weights for importance sampling. This is because when $\mathbf{x}$ is within the bounding box of a child node, $d_j^{\min}(\mathbf{x})$ becomes zero and the weight becomes infinity. When $\mathbf{x}$ is outside of the bounding box of the other child node, the probability of selecting this other child node becomes zero, regardless of how much illumination it represents. Therefore, this sampling scheme would not necessarily converge to the correct result.

A typical solution to this problem would be replacing the closest distance $d_j^{\min}(\mathbf{x})$ with the distance to a point within the bounding box of a node. Indeed, this is the approach used by adaptive tree splitting [EK18] by computing the distance to the centroid of the bounding box. However, regardless of how this point is chosen, the singularity of inverse-square distance to a point, where the weight would go to infinity, would not be entirely avoided. Also, picking a particular point within the bounding box may lead to a weight that can be a poor estimator of the incoming illumination from within the entire bounding box, resulting in high noise and low convergence. In effect, this solution reduces the singularity in Equation 1 from an entire bounding box to a point, but does not always improve the convergence of importance sampling, as compared to merely considering the total light intensities [SWZ96].

On the other hand, we cannot completely ignore the inverse-square attenuation term, as it can be a significant factor in bounding the illumination contribution of distant lights. Therefore, simply considering the total light intensity $\mathbf{I}_j$ or its modulation with the reflectance bound $F_j(\mathbf{x},\omega)$ are not ideal solutions either.

Our solution is simply combining the weights computed using the minimum distance term $d_j^{\min}(\mathbf{x})$ and weights computed without a distance term. At the higher levels of the hierarchy, where the node bounding boxes are large and the minimum distance term can be a poor indicator of the expected illumination coming from a node, we simply ignore the distance term. For lower levels of the hierarchy, where the node bounding boxes are sufficiently far from $\mathbf{x}$, we include the distance term. More precisely, we decide whether to include the distance term by comparing the minimum distance $d_j^{\min}(\mathbf{x})$ to the size of the bounding box ℓ_j (i.e. the length of its diagonal). The distance term is included only if $d_j^{\min}(\mathbf{x}) > \alpha\ell_j$ for both child nodes, where α is a user-defined scaling coefficient. We use $\alpha = 1$ for all examples in this paper. Let j and k indicate two child nodes of an internal node. We compute the weights using

$$w_j = F_j(\mathbf{x},\omega)\left\|\mathbf{I}_j\right\|\Lambda_{j,k}(\mathbf{x})\,, \tag{2}$$

where the attenuation term $\Lambda_{j,k}(\mathbf{x})$ is

$$\Lambda_{j,k}(\mathbf{x}) = \begin{cases} \frac{1}{\left(d_j^{\min}(\mathbf{x})\right)^2} & \text{if } d_j^{\min}(\mathbf{x}) > \alpha\ell_j \text{ and } d_k^{\min}(\mathbf{x}) > \alpha\ell_k \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

For incorporating directional lights with no attenuation factor, we slightly modify the weight computation, such that

$$w_j = F_j(\mathbf{x},\omega)\left(\left(\left\|\mathbf{I}_j\right\| - \left\|\mathbf{I}_j^D\right\|\right)\Lambda_{j,k}(\mathbf{x}) + \left\|\mathbf{I}_j^D\right\|\right)\,, \tag{4}$$

where $\left\|\mathbf{I}_j^D\right\|$ is the total intensity of directional lights within the node. Thus, the attenuation term does not impact the importance of directional lights.

3.2.2. Dead Branches

We use the term *dead branch* to refer to the subtree under a node that can contribute no illumination purely due to the reflectance bounds of all lights within the subtree (not considering visibility/shadows). We call all nodes within a dead branch *dead nodes*.

When the reflectance bound $F_j(\mathbf{x},\omega)$ for the root node of a dead branch is zero, the node is assigned zero probability and the dead branch is automatically avoided during our hierarchical importance sampling traversal. However, this behavior is not guaranteed. This is because a typical computation of the reflectance bound would be based on a conservative estimate, using the bounding box of the light node [WFA*05, EK18], so a dead branch can be assigned a non-zero probability.

Including dead branches in lighting estimation does not break the lightcuts algorithm, especially when it is permitted to converge using as many light samples as necessary to satisfy the error threshold, but it introduces unnecessary computation cost. More importantly, when the maximum light sample count is limited, selecting a dead branch can effectively *waste* a light sample and negatively impact the quality of the lighting estimation. This is particularly important for fast lighting estimation with a small number of samples using algorithms that rely on multi-sampling, such as path tracing. Therefore, dead branches should be avoided when detected.

A dead branch is automatically detected and avoided when $F_j(\mathbf{x},\omega) = 0$ at its root node. Otherwise, we detect a dead branch further down its subtree, where the importance weights w_1 and w_2 of the two child nodes of a node are both zero. Note that testing whether $w_1 + w_2$ is zero is sufficient for detecting a dead branch, because all possible subtrees within a dead branch include a node with $w_1 + w_2 = 0$, since all leaf nodes of a dead branch contain lights i with $F_i(\mathbf{x},\omega) = 0$. However, the node that satisfies this condition does not have to be the root node of the dead branch.

Once a dead branch is detected, we can simply terminate the hierarchical traversal by returning any light within the subtree. The selection of a light within a dead branch is inconsequential, since none of them should have any effective illumination. Therefore, we can simply return the representative light of an internal node and skip the evaluation of the light source to save computation.

Yet, simply terminating the traversal and returning an arbitrary light within the subtree of a dead branch would still waste a light sample. Therefore, our solution is backtracking the hierarchical traversal and making sure that we do not select a light within a dead branch. This can be done by moving back up the light tree, skipping the detected dead node by updating its probability to zero, and continuing the traversal using a different path down the light tree. Note that in some cases the entire light tree can be a dead branch (i.e. a *dead tree*) if the point $\mathbf{x}$, where the illumination is computed, cannot be illuminated by any light in the scene (based on the reflectance bound alone, not considering visibility). *Note that backtracking to avoid dead branches introduces bias in light sampling.*

3.3. Implementation

The lightcuts algorithm determines a "cut" through the light tree by evaluating it starting from its root node. Each evaluated light tree node is added to a light sample list (typically implemented using a heap) to be returned. The nodes in the light sample list with an error bound greater than the error threshold are replaced by their child nodes (in the order of the largest error bound). When replacing a node with its child nodes, only one of the child nodes requires a full evaluation, while the other one (containing the light sample of the parent node) uses the parent node's data and only updates its error bound and cluster intensity.

The implementation of the stochastic lightcuts algorithm involves only two relatively minor changes. First, instead of simply using the representative light of a light tree node, we use hierarchical importance sampling to select a light within the subtree of the node. If the node is determined to be a dead node, we simply skip adding the node to the light sample list. If the root node is a dead node, we return an empty light sample list. Second, each light sample returned must also be accompanied by the probability of selecting the light sample within the subtree of the selected node. Therefore, the light selection probabilities must be updated when replacing a node in the light sample list with its child nodes.

4. Results

We evaluate our stochastic lightcuts method by first comparing it to lightcuts and then comparing its sampling quality to other

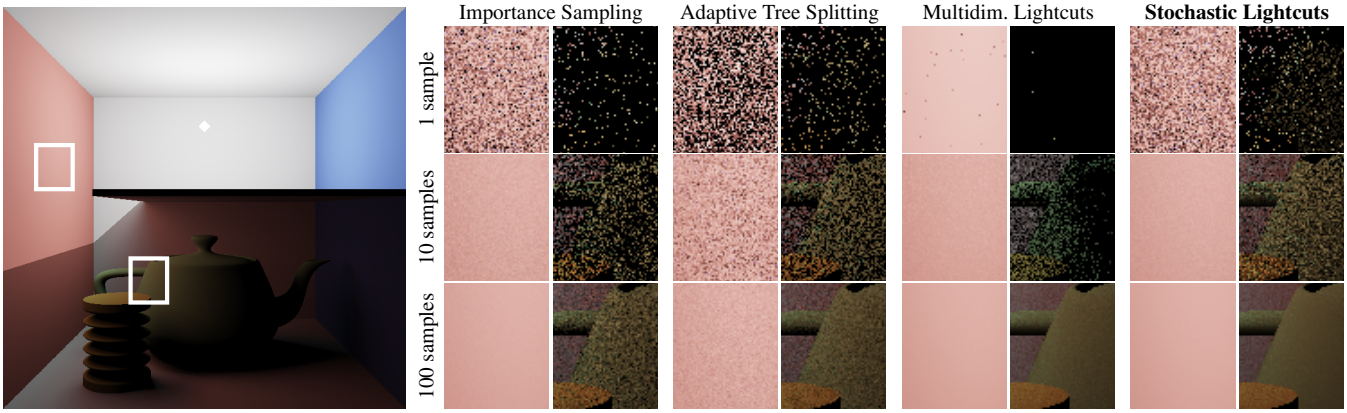

Figure 2: *Comparison of the sampling quality of our stochastic lightcuts method to other related sampling techniques, rendered with a single sample per pixel using 1, 10, and 100 light samples for lighting estimation. Traditional importance sampling [SWZ96] produces a relatively low-noise image in well-lit areas (left column), but the noise increases in darker areas (right column). Adaptive tree splitting [EK18] provides improved sampling quality for darker regions (right column), but introduces more noise in well-lit areas (left column), as compared to traditional importance sampling. Multidimensional lightcuts [WABG06] produces low noise in well-lit areas (left column), but sampling quality degrades substantially in darker regions (right column) and the results are temporally unstable. Our stochastic lightcuts method produces the best sampling quality in all cases.*

related stochastic sampling techniques. The results are generated with a custom renderer using Intel's Embree ray tracing kernels [WWB*14]. The error threshold for lightcuts and stochastic lightcuts is set to 2%. All timing results are measured on a computer with dual Intel Xeon CPUs running at 2.4 GHz (16 total cores).

4.1. Comparison to Lightcuts

We compare the results of lightcuts and stochastic lightcuts using three different scenes. Figure 1 shows three different scenes with three different rendering methods: direct illumination only, path tracing, and virtual spherical lights [HKWB09].

We show two different settings for lightcuts. The first one limits the maximum number of light samples to 10. This causes the lightcuts algorithm to produce a fast estimate of lighting, but this relatively small limit on light samples does not allow lightcuts to converge using its error threshold parameter. Therefore, the resulting images contain large blocks of dark areas, randomized shadow patterns, and inconsistent illumination throughout the scenes with sharp illumination variations.

The second setting for lightcuts allows the algorithm to converge using its error threshold. In this case, we limit the maximum light samples to 1000, which is close to brute-force lighting evaluation in the first two scenes. However, lightcuts does not come close to this maximum limit and typically returns an estimation using only a few hundred light sources. Nonetheless, the performance gain over brute force computation is limited, as compared to using much fewer light samples. Moreover, the resulting solution still contains sampling correlation that leads to visible flickering in animations.

Our stochastic lightcuts solution, in comparison, provides a fast lighting estimation with low noise, using a maximum of 10 light samples per lighting estimation. Since there is no sampling correlation with stochastic lightcuts, the results do not contain visual artifacts and it is temporally stable, using a different light tree at each frame. The stochastic lightcuts algorithm takes considerably more time than lightcuts with the same number of samples, due to the cost of hierarchical traversal of the light tree down to leaf nodes. Also, sampling correlation, while detrimental to quality, improves performance with better cache utilization and leads to shorter render times using lightcuts with the same number of light samples. Nonetheless, using only 10 light samples, we can provide a solution that is similar to the lightcuts solution with up to 1000 light samples, and without its temporal instability. The render times are also about an order of magnitude shorter. It should be noted that multi-sampling used in these scenes help reduce the resulting noise with stochastic lightcuts, but does not help lightcuts at all.

4.2. Stochastic Sampling Comparisons

We evaluate the sampling quality of our stochastic lightcuts method by comparing it to traditional importance sampling [SWZ96], adaptive tree splitting [EK18], and multidimensional lightcuts [WABG06]. We present the results of an experiment in Figure 2, using a simple scene with 10,000 virtual lights. We use only a single sample per pixel to clearly show the quality of light sampling. Using multiple samples (i.e. multiple lighting estimations) per pixel reduces the noise for all methods but multidimensional lightcuts, which receives a relatively minor improvement from multi-sampling due to sampling correlation.

In this scene, traditional importance sampling [SWZ96] provides a good sampling quality, particularly for the top half of the image. The bottom half, however, receives more noise because of the high probability of sampling the original light, which is shadowed, and the approximately equal probability of sampling any virtual light.

Adaptive tree splitting [EK18] improves the sampling quality for the darker bottom region of the image, as compared to traditional importance sampling. On the other hand, the singularity in its im-

portance formulation leads to poor importance estimates for the top half of the image. As a result, the strong original light source is not sampled as often and the importance estimates considerably deviate from the illumination contributions of the selected lights, leading to even more noise than purely considering light intensities, like traditional importance sampling [SWZ96].

The multidimensional lightcuts method [WABG06] augments lightcuts by storing 32 representative lights per light tree node. One of these representative lights is randomly selected (using proportional probabilities to light intensities) during the lighting estimation of the node. Thus, it introduces some flavor of stochastic sampling in lightcuts, but the benefits are highly limited. It preserves the excellent convergence rates of lightcuts, such that the representative lights, chosen during the light tree construction, are likely to produce low error. On the other hand, sampling correlation is not entirely avoided, so the results display a similar temporal instability to lightcuts. In this test scene, it provides excellent results for the top half, but the bottom half is either completely dark (with 1 light sample) or extremely noisy and unstable (with 10 light samples). Using 100 light samples virtually eliminates the noise, but the temporal instability persists.

Our stochastic lightcuts method provides the best sampling quality. Using a single light sample, it slightly improves the sampling quality of traditional importance sampling in the top half of the image, and it achieves a more significant improvement with more light samples. This improvement is due to the importance estimation of stochastic lightcuts, which locally-adaptive and provides improved accuracy in importance estimation, considering the light distances from the point where lighting is evaluated. In the bottom half, it leads to a superior sampling quality than all other methods. Like multidimensional lightcuts, it preserves the excellent convergence rate of lightcuts when using a relatively large number of light samples, but it does not suffer from temporal instabilities.

5. Discussion and Future Work

While algorithmically similar to lightcuts, our stochastic lightcuts solution is conceptually closer to traditional importance sampling [SWZ96]. The light tree is mainly used for efficiently computing importance sampling weights for lights. Therefore, stochastic lightcuts is safe to use with an arbitrarily small number of maximum light samples (such as one), though adding more light samples reduces the noise faster than traditional importance sampling.

Since the light tree is mainly used for computing the importance weights, the construction of the light tree impacts the quality of the sampling scheme. Our experiments with stochastic lightcuts revealed that using the same set of light sources, different light trees lead to different convergence rates at different parts of the image. Indeed, a poorly constructed light tree means poor importance sampling weights. In all our tests, we use the light tree construction algorithm of the original lightcuts method [WFA*05], which consistently provides superior sampling quality for stochastic lightcuts, as compared to prior methods [SWZ96, WABG06, EK18]. However, we have also noticed a more significant variation in sampling noise with our method, depending on the construction of the light tree. This suggests that the results of stochastic lightcuts can be further improved by a specialized light tree construction algorithm, which would be an interesting direction for future work. Note that unlike the original lightcuts method that must use a stochastic light tree construction algorithm to avoid introducing bias during its deterministic lighting estimation, stochastic lightcuts can use a deterministic light tree construction algorithm, as it would not introduce bias during rendering with its stochastic lighting estimation.

An important advantage of stochastic lightcuts is that it allows achieving a fast lighting estimation using a very small number of light samples. This is unlike any prior scalable lighting solution [WFA*05, WABG06, HPB07, YY17, EK18], all of which require a large number of samples in practice.

6. Conclusion

We have presented the stochastic lightcuts method, which introduces stochastic sampling into the lighting estimation of lightcuts. We have also presented a robust hierarchical importance sampling approach for improving the estimation, especially when used with a small number of light samples. Our approach eliminates the sampling correlation problems of lightcuts that lead to temporal instabilities and allows incorporating different light types, making our solution suitable for a much wider range of applications. Furthermore, because we can effectively estimate the illumination using a small number of light evaluations, we can achieve more than an order of magnitude faster lighting estimation than lightcuts.

References

[DKH*14] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable realistic rendering with many-light methods. *Computer Graphics Forum 33*, 1 (2014), 88–104. 2

[EK18] ESTEVEZ A. C., KULLA C.: Importance sampling of many lights with adaptive tree splitting. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of HPG 2018) 1*, 2 (2018), 25:1–25:17. 3, 4, 5, 6

[HKWB09] HAŠAN M., KŘIVÁNEK J., WALTER B., BALA K.: Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia 2009) 28*, 5 (2009), 143:1–143:6. 5

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2007) 26*, 3 (2007). 2, 6

[LY19] LIN D., YUKSEL C.: Real-time rendering with lighting grid hierarchy. *Proc. ACM Comput. Graph. Interact. Tech. (Proceedings of I3D 2019) 2*, 1 (2019), 8:1–8:17. 2

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics 15*, 1 (1996), 1–36. 2, 3, 5, 6

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) 25*, 3 (2006), 1081–1088. 2, 5, 6

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005) 24*, 3 (2005), 1098–1107. 2, 3, 4, 6

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics 33*, 4 (2014), 143:1–143:8. 5

[YY17] YUKSEL C., YUKSEL C.: Lighting grid hierarchy for self-illuminating explosions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017) 36*, 4 (2017), 110:1–110:10. 2, 6