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

Dynamic Many-Light Sampling for Real-Time Ray Tracing

P. Moreau[1,2] , M. Pharr[1] and P. Clarberg[1]

[1]NVIDIA
[2]Lund University, Sweden

Abstract

Monte Carlo ray tracing offers the capability of rendering scenes with large numbers of area light sources—lights can be sampled stochastically and shadowing can be accounted for by tracing rays, rather than using shadow maps or other rasterization-based techniques that do not scale to many lights or work well with area lights. Current GPUs only afford the capability of tracing a few rays per pixel at real-time frame rates, making it necessary to focus sampling on important light sources. While state-of-the-art algorithms for offline rendering build hierarchical data structures over the light sources that enable sampling them according to their importance, they lack efficient support for dynamic scenes. We present a new algorithm for maintaining hierarchical light sampling data structures targeting real-time rendering. Our approach is based on a two-level BVH hierarchy that reduces the cost of partial hierarchy updates. Performance is further improved by updating lower-level BVHs via refitting, maintaining their original topology. We show that this approach can give error within 6% of recreating the entire hierarchy from scratch at each frame, while being two orders of magnitude faster, requiring less than 1 ms per frame for hierarchy updates for a scene with thousands of moving light sources on a modern GPU. Further, we show that with spatiotemporal filtering, our approach allows complex scenes with thousands of lights to be rendered with ray-traced shadows in 16.1 ms per frame.

CCS Concepts

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

1. Introduction

Complex illumination is a critical ingredient for the visual richness of rendered images. Images that include the soft shadows and diffused lighting that is characteristic of large area light sources have a markedly more realistic appearance than images rendered with small numbers of point or directional light sources, which give stark and harsh lighting effects. However, with more than few light sources it is infeasible to shade them all individually, especially under the constraints of real-time rendering. Culling and/or stochastic selection of a subset of lights is necessary. In this work, we focus on stochastic sampling in order to be able to support many contributing light sources and still compute unbiased results.

With this approach, it is necessary to define a discrete probability density function (PDF) $p_l(\mathbf{x}, i)$ that gives the probability of sampling the ith light as seen from a point $\mathbf{x}$ in the scene. The more closely proportional $p_l(\mathbf{x}, i)$ is to the reflected light at $\mathbf{x}$ due to the light i's emission, the less error will be present in the image. Unfortunately, an accurate p_l cannot be easily precomputed as there are millions of shading points $\mathbf{x}$, a scene may have tens of thousands of lights i, and generally, the optimal sampling distribution varies drastically between different parts of the scene.

An elegant solution exists for offline rendering: a single *bounding volume hierarchy* (BVH) is built over all lights and at each point $\mathbf{x}$, the tree is stochastically traversed [KWR*17, CEK18]. At each level of the traversal, the relative contributions of the children nodes are estimated such that the full distribution p_l is never represented explicitly and only $\log(n)$ computations per shading point (where n is the number of lights) are required. This idea is illustrated in Figure 1. For offline rendering, the cost of constructing the light BVH is negligible compared to the rendering time. This is not the case for real-time rendering, where many fewer rays are generally traced per frame and no more than a few milliseconds per frame are available. The goal of this paper is to adapt light BVH methods to be suitable for real-time ray tracing of dynamic scenes. We make the following contributions:

- We organize light sources in multiple bounding volume hierarchies, arranged in a two-level hierarchy.
- We show that refitting light BVHs without modifying their topology can be implemented efficiently on the GPU, and that this approach works well for moderate amounts of light source motion.
- We demonstrate that top-level BVHs can be rebuilt asynchronously to maintain close-to-optimal overall tree topology. Thus, our approach can support a wide range of motion without an increase in sampling error due to sub-optimal BVHs.
- We present results based on a real-time path tracer implemented in Direct3D 12 using the DirectX Raytracing API.

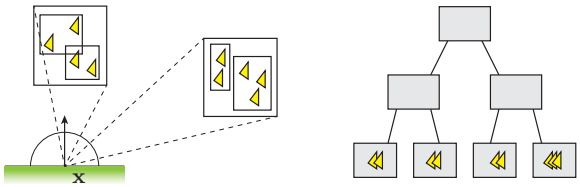

Figure 1: *Light sources are stored in a bounding volume hierarchy (illustrated in 2D on the left, and as a tree on the right). To sample a light at a shading point* **x***, the tree is stochastically traversed by estimating the contributions from the two children (light clusters) at each node. Important clusters are given a higher priority and a random decision is made about which branch to follow.*

We show that with our approach, the cost of maintaining the light acceleration structure is less than 1 ms per frame for dynamic scenes with tens of thousands of emitters, with mean-squared error (MSE) within 6% compared to a single optimized light BVH.

2. Previous Work

Kajiya suggested taking a single light sample at each path vertex regardless of the total number of lights [Kaj86]. Since his work, a number of researchers have investigated ways of computing accurate estimates of the contributions of the light sources to be able to choose among them more effectively.

Ward introduced the idea of generating a discrete PDF over lights at each shaded point [War91] and tracked how often each light source was visible. Shirley et al. estimated lights' contributions to compute per-light probabilities [SWZ96], using an octree to classify light sources into "bright" and "dim" sets. Zimmerman and Shirley used a uniform spatial subdivision rather than an octree and also maintained estimates of each light's visibility [ZS95]. Wald et al. generated a light sampling PDF using a sparse path tracing pass [WBS03]. Their approach works well in densely occluded environments but does not effectively distinguish between local lights that are important at some points but less so at others.

A number of light transport algorithms have been developed based on hierarchical representations of illumination encoded as point lights, including Lightcuts [WFA*05] and its predecessor [PPD98]. Closely related are global illumination algorithms based on virtual point lights (VPLs) [DKH*14]. These approaches all use point lights for illumination and not just for sampling, which introduces the possibility of error from the discretization and the weak singularity from the $1/r^2$ term close to the point lights.

Tiled shading [And09, OA11, Har12, OC17] bins lights into screen-space tiles, where the depth bounds of the tiles reduce the number of lights that need to be processed in each one. These screen space acceleration structures are not applicable to indirect intersection points with ray tracing. Further, clamped light ranges can cause undesired darkening. To address the darkening, Tokuyoshi and Harada used a bounding sphere hierarchy and stochastic light ranges to reject unimportant lights [TH16].

A number of researchers have investigated other approaches based on building hierarchies over the light sources and traversing them to sample lights. Iray uses a hierarchical light importance sampling scheme based on a BVH [KWR*17]. Conty Estévez and Kulla [CEK18] take a similar approach for cinematic rendering with a 4-wide BVH that clusters lights in world space based on both orientation and surface normal bounding cones. Moreau and Clarberg describe a GPU implementation of their algorithm [MC19]. Vévoda et al. [VKK18] recently described an approach based on online learning of the importance of light sources based on clustering with a hierarchy and a Bayesian approach. Their method has relatively high memory use, does not support dynamic light sources, and does not readily map to GPUs.

The idea of "refitting" bounding volume hierarchies for ray intersection acceleration with animated geometry was introduced by van den Bergen [vdB97] for collision detection, and later applied to ray–object intersection [LYMT06, WBS07]. These approaches take advantage of the fact that for relatively small amounts of object motion, the original BVH's topology can be maintained with node bounds updated to account for moving objects' new positions. Doing so saves the computational expense of rebuilding the BVH. To our knowledge, these techniques have not been applied to light sampling BVHs. We note that with collision detection and ray–object intersection, sub-optimal BVHs cause an increase in computation but do not introduce error. With light sampling, low quality BVHs instead lead to inaccurate light contribution estimates, which in turn may lead to variance in rendered images—i.e., error rather than inefficiency. We apply refitting and study these effects.

3. Algorithm

The light sampling distribution should ideally be proportional to each light i's contribution to reflected radiance L at the point $\mathbf{x}$ being shaded: $p_l(\mathbf{x}, i) \propto L(\mathbf{x}, \omega_o)$, where ω_o is the view direction. The more closely the distribution matches L, the lower the error will be. This principle is known as *importance sampling* in Monte Carlo integration [PJH16]. However, it is not feasible to determine a p_l that is exactly proportional to L:

$$L(\mathbf{x}, \omega_o) = \int_{A_i} \frac{f(\omega \to \omega_o)\, L_e(\mathbf{x}_l, -\omega)\, V(\mathbf{x} \leftrightarrow \mathbf{x}_l)\, |\cos\theta \cos\theta_l|}{\|\mathbf{x} - \mathbf{x}_l\|^2} \, d\mathbf{x}_l, \tag{1}$$

where the integration domain A_i is the surface of the ith light, f is the bidirectional scattering distribution function (BSDF), ω is the normalized vector from $\mathbf{x}$ to a point $\mathbf{x}_l$ on the light, L_e is the radiance emitted by the light, and V is a visibility term that is one if the two points are unoccluded and zero otherwise. The two cosine terms are with respect to the surface normals at $\mathbf{x}$ and at $\mathbf{x}_l$.

We use the approach by Conty Estévez and Kulla [CEK18] as implemented by Moreau and Clarberg [MC19], which takes the total emitted flux, the $1/r^2$ falloff, and the relative orientations of the shading normal and light source into account using bounding cones. Instead of computing these terms for all the light sources, lights with nearby locations and directions are grouped together into a light BVH [KWR*17,CEK18]. The BVH allows hierarchical approximation of these quantities, reducing the per-sample complexity from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$ and making it feasible to perform these computations at every shading point. This algorithm can be applied to point or area lights, as well as emissive triangles.

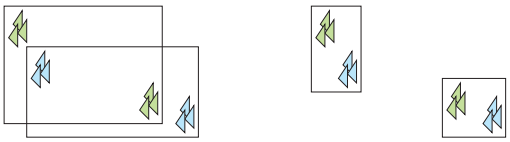

Figure 2: *Care has to be taken when deciding which light sources to put in each bottom-level hierarchy. Left: it is difficult to accurately estimate the contribution from the large overlapping BLASes, leading to higher variance. Right: it is preferable to place lights that are spatially nearby in the same hierarchies.*

3.1. Two-Level Light Acceleration Structures

Previous approaches used a single BVH for all light sources. The BVH must be rebuilt from scratch if even a single light moves or changes intensity. That approach isn't suitable for real-time rendering with dynamic lights due to the cost of rebuilding the BVH [MC19].

The problem is closely related to managing data structures for ray-intersection testing in dynamic scenes. APIs like DirectX Raytracing or Vulkan ray tracing and libraries like OptiX [PBD*10] and Embree [WWB*14], use *two-level BVHs* that store collections of geometry in separate bottom-level acceleration structure (BLAS) and maintain a separate top-level acceleration structure (TLAS) that stores the BLASes. A moving object only causes its BLAS and the TLAS to be rebuilt. The cost of hierarchy updates is kept low if static geometry is stored separately from dynamic objects.

With light BVHs, that partitioning is not ideal for static lights as it would lead to large BVH nodes and therefore poor estimates of node contributions due to having many emissive primitives and high uncertainty regarding their positions and orientations within the node. Other strategies such as sorting based on material can similarly be counterproductive; see Figure 2. We have found that storing each emissive mesh in its own BLAS generally gives a good balance. Figure 3 shows an example from one of our test scenes. In future work, it would be interesting to investigate automatic ways to partition emissive geometry into BLASes.

We sample our two-level light BVH by first traversing the TLAS down to a leaf node by evaluating an importance function [CEK18] for each of the current node's children and stochastically selecting one of them. Each leaf node points to a BLAS, and the same technique is used to select a light in it. The overall probability of sampling a light is the product of the probability of sampling the BLAS it is in and the probability of sampling it in its BLAS.

3.2. Updating the Two-Level Acceleration Structure

By design, if a light source has been modified then only the TLAS and BLAS to which it belongs need to be updated. Those updates can either take the form of fully rebuilding the hierarchy or refitting it. The latter keeps the topology of the hierarchy intact—i.e., the parent/children relationships between the nodes stay the same, as well as which primitives are stored in each leaf node—but the aggregate attributes like bounding boxes are updated to account for the object motion [LYMT06, WBS07]. For our light BVHs, the aggregate attributes also include the total emitted flux and normal bounding cones. Thus, even static light sources can trigger a rebuild or refit if their flux changes (for example, flashing lights).

Figure 3: *Each yellow box is the root node of a bottom-level acceleration structure (BLAS), here shown for the Bistro scene. Notice that both static light sources, e.g., the street lights and hanging light bulbs, and dynamic emissive objects are each represented by one or more BLASes, here in total 142 for the full scene.*

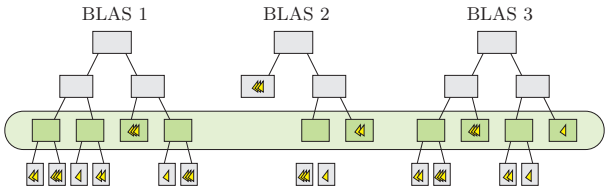

Figure 4: *Our light BVH refit operation is performed bottom-up on all modified bottom-level hierarchies in parallel, with one compute shader dispatch per tree level (as shown in green). For this purpose, each tree stores a list of node indices sorted by tree level.*

After object animation, we dispatch compute shader passes on the GPU for all modified BLASes, with a separate dispatch for each tree level bottom-up, updating the current row's nodes based on their children's aggregate attributes; see Figure 4. We then also refit the TLAS on the GPU, to let rendering immediately proceed. However, we have found it worthwhile to rebuild the TLAS to keep it as accurate as possible. Therefore, we perform an asynchronous full rebuild of the TLAS on the CPU; this lets us exploit idle CPU cycles while the GPU is busy rendering without introducing extra latency. Figure 5 shows an execution timeline.

4. Results

We evaluated our approach using two complex scenes with dynamic light sources; see below for general statistics and Figure 3 and 6 for representative images. The scenes contain both skinned animation and multiple rigid objects that follow animation paths. Note that we perform BVH refit for all moving objects, independent of their amount of motion and type of animation. The algo-

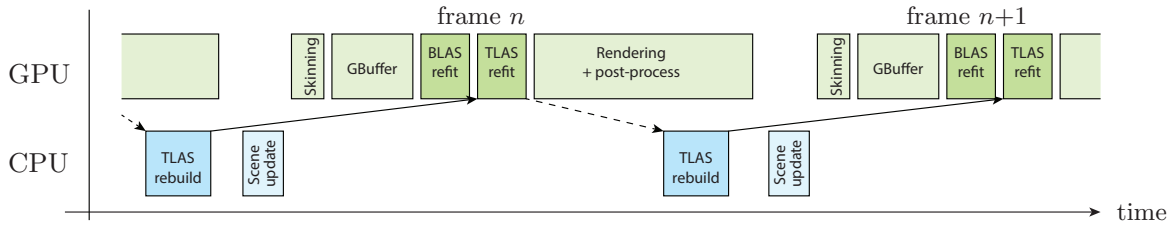

Figure 5: *The bottom-level (BLAS) and top-level (TLAS) light BVHs are refitted on the GPU based on the current vertex positions and flux values for the emissive geometry. To maintain a good topology, the TLAS is rebuilt asynchronously on the CPU based on last frame's data.*

rithms were implemented in the Falcor real-time rendering framework [BYF*18] using Microsoft Direct3D 12 and DirectX Raytracing (DXR). All results were measured on an NVIDIA GeForce RTX 2080Ti GPU with 12GB RAM using driver 419.67, on a computer with an Intel Xeon E5-1650 v4 CPU at 3.60GHz and 32GB RAM. The output resolution was 1920×1080 pixels.

Scene	Bistro	Emerald Square
Total triangles	3,038,170	9,687,074
Emissive triangles (static)	19,948	19,440
Emissive triangles (dynamic)	6,495	66,172

Using a two-level BVH rather than a single unified one may reduce importance sampling accuracy (recall Figure 2.) In order to evaluate the importance of this issue, we first measured rendering time and the mean squared error (MSE) for static scene using one light sample per pixel (spp) with three sampling approaches: uniform probabilities over all of the light sources, a single light sampling BVH on the GPU [MC19], and our two-level BVH. Error was measured with respect to reference images rendered with 10,000 spp. Times were averaged over a few hundred frames and only include the ray-tracing pass to compute lighting—the time to generate the G-buffer (roughly 2 ms) and for tone mapping (less than 1 ms) is not included. All techniques were combined with BRDF sampling using multiple importance sampling (MIS) [VG95], taking one BRDF sample for each light sample. For clarity of presentation, only direct illumination was evaluated; the total number of rays per pixel was therefore 2×spp (one shadow ray and one BRDF scatter ray).

The results are presented in Table 1. As has been demonstrated previously [CEK18, MC19] and is evident here, a uniform light sampling PDF is ineffective in scenes with many light sources. For these scenes, the increase in MSE from replacing a single BVH with a two-level BVH is insignificant. Note also that there is a negligible difference in runtime performance between these variants.

We next performed a set of experiments to compare four approaches for rendering scenes with moving light sources: uniform light sampling; a single-level BVH that is built from scratch when any light changes; a single-level BVH that is refit when a light changes; and a two-level BVH where bottom-level BVHs are refit and the top-level is rebuilt. Lacking an efficient GPU algorithm to build the entire BVH, the second approach is not suitable for real-time applications—a full rebuild for these scenes takes over 90 ms—but it provides a baseline that gives the best sampling probabilities and thus the lowest MSE.

Because all of these sampling methods are unbiased, *Monte Carlo efficiency* is a useful metric. It is defined as the inverse product of rendering time t and variance v [PJH16], which MSE is an estimate of: $\varepsilon = \frac{1}{t \cdot v}$. Monte Carlo efficiency cleanly accounts for the interplay between computation time and error in the integration: because variance (and MSE) decreases at the rate $\mathcal{O}(1/N)$ for a number of samples N, it correctly indicates that, for example, a method that takes twice as much time as another to deliver half as much variance is no improvement: we could equivalently take twice as many samples with the first and expect the same variance.

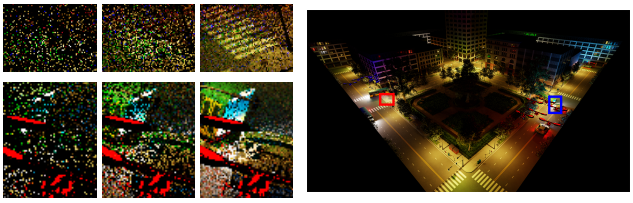

Figure 6: *The Emerald Square scene with crops rendered using 1 spp, 4 spp, and 16 spp, respectively. This scenes has 143 dynamic emissive meshes for a total of 66,172 moving emissive triangles.*

The results are reported in Table 2. As before, timings are the average over a few hundred frames and MSE is computed with respect to a reference rendering with 4,000 spp. We can see that the slight reduction in update time from refitting a single BVH is not worth it: the increase in MSE is such that its Monte Carlo efficiency is on par with or lower than our approach. In a similar fashion, we can see that although a single-level BVH that is built from scratch when lights move gives the best MSE, the cost to build the BVH also is not worth it in terms of overall efficiency.

Next, we measured the effect of the accumulation of error from refitting BVHs over long animations—extensive motion of light sources may cause the original BVH topology to become inappropriate. See the accompanying video in order to see the animation. The results are presented in Figure 7 and 8. We can see that rebuilding the TLAS is worth the small computational cost.

Finally, as presented thus far, our method produces consistent, unbiased results. For practical use at low sample counts, it is essential to pair it with a reconstruction filter to remove the residual Monte Carlo noise. As a proof of concept, we have integrated *spatiotemporal variance-guided filtering* (SVGF) [SKW*17] in the renderer. Figure 9 shows an example rendered at 1 spp using uniform sampling and our method, before and after denoising. Note that the denoised result with our method is much closer to ground truth, thanks to the denoiser having better input to work with. Our

Scene	Bistro			Emerald Square		
	Uniform	One-level BVH	Two-level BVH	Uniform	One-level BVH	Two-level BVH
Time (ms)	6.2	10.4	10.7	7.7	11.3	11.6
MSE	16.8	2.12	2.14	19	0.49	0.50

Table 1: *Rendering time and error using 1 spp for the first frame of the animations of the two scenes with time paused, i.e., no dynamic updates were performed. The two-level BVH only introduces a negligible increase in error of 1–2% compared to a single-level BVH.*

Scene	Bistro				Emerald Square			
	Uniform	One-level (rebuild)	One-level (refit)	Two-level (rebuild/refit)	Uniform	One-level (rebuild)	One-level (refit)	Two-level (rebuild/refit)
BVH update time (ms)	0	~90	0.17	0.85 / 0.18	0	~300	0.22	0.89 / 0.35
Sampling time (ms)	0.34	2.3	2.4	2.6	0.32	2.0	2.0	2.2
Total time (ms)	6.2	101	10.8	12.0	7.7	311	11.3	12.6
MSE	16.5	1.56	1.95	1.65	20	0.58	0.67	0.61
MC efficiency ε	0.0097	0.0064	0.048	0.050	0.0065	0.0055	0.13	0.13
ε w.r.t. uniform	1×	0.66×	4.9×	5.2×	1×	0.85×	20.3×	20.1×

Table 2: *Performance and error for with two different scenes rendered at 1 spp. Error was measured after 269 frames in order to capture large amounts of light movement. By comparing the Monte Carlo efficiency of the various approaches, we can see that our approach (two-level) has a much higher efficiency than one-level rebuild, without suffering from the robustness issues of refitting only, as pictured in Figure 8. Notice that we have summed the CPU and GPU execution times in this table, even though in reality they overlap.*

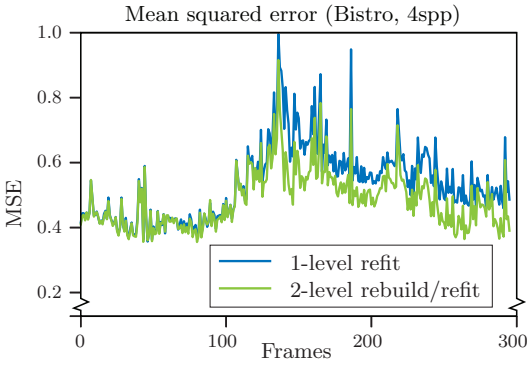

Figure 7: *Error over time for an animation of the Bistro scene rendered at 4 spp. The benefit of rebuilding the top-level BVH results in an up to 1.4× reduction in MSE compared to using a one-level BVH that is refitted every frame. Notice that locally the differences can be much larger; see Figure 8.*

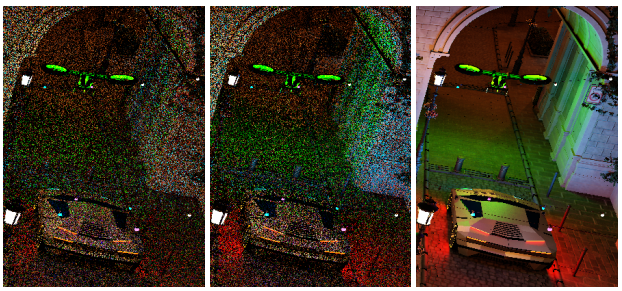

Figure 8: *Crops of the Bistro scene for frame 269 of the animation at 4 spp, using a one-level BVH that is refitted each frame on the left, our two-level BVH in the center, and a reference on the right.*

SVGF implementation has not been optimized, and currently runs in roughly 4 ms per frame, for a total frame time of 16.1 ms for this scene. The supplemental video shows SVGF filtering with animation. The video was rendered with 4 spp, giving a frame time of 31.7 ms. Note that the filtered output is generally of high quality, with only minor temporal artifacts and ghosting. Gradient estimation [SPD18] would presumably reduce the ghosting.

5. Conclusion and Future Work

The arrival of ray tracing as a first-class visibility primitive in modern graphics APIs presents an opportunity for substantial improvements in the realism and richness of real-time graphics. We have introduced an approach for unbiased many-light sampling on GPUs that is suitable for ray tracing dynamic scenes. Our method is based on a two-level hierarchy of acceleration structures over the lights and includes efficient update and sampling algorithms.

In the future, we would like to develop algorithms for building light BVHs from scratch on the GPU. Not only would it be desirable to eliminate CPU-GPU copies, but a sufficiently efficient algorithm would also allow the option of building a single BVH from scratch for moderate numbers of dynamic light sources. We would also like to investigate heuristics for determining when a refit light BVH has come to be ineffective and should be rebuilt. We hope that our work will inspire forward looking game engines and further research in rendering with complex lighting.

Acknowledgments Thanks to Nicholas Hull and Kate Anderson for creating the scenes. The Bistro scene is based on assets kindly donated by Amazon Lumberyard. The car model was made by Turbosquid user barteks2, and the helicopter asset by Sketchfab user f3nix. We would also like to thank Lund University, Aaron Lefohn and NVIDIA Research for supporting this work, and the Swedish Research Council for funding Pierre under grant 2014-5191.

Figure 9: *Crops of the* Bistro *scene and MSE with respect to uniform light sampling. From left to right: uniform light sampling at 1 spp; filtered with SVGF [SKW*17] (5 frames accumulated); two-level BVH sampling at 1 spp, filtered output (5 frames acc.); reference image.*

References

[And09] ANDERSSON J.: Parallel Graphics in Frostbite—Current & Future. Beyond Programmable Shading, SIGGRAPH Courses, 2009. 2

[BYF*18] BENTY N., YAO K.-H., FOLEY T., OAKES M., LAVELLE C., WYMAN C.: The Falcor rendering framework, 05 2018. URL: https://github.com/NVIDIAGameWorks/Falcor. 4

[CEK18] CONTY ESTÉVEZ A., KULLA C.: Importance Sampling of Many Lights with Adaptive Tree Splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 2 (2018), 25:1–25:17. 1, 2, 3, 4

[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

[Har12] HARADA T.: A 2.5D Culling for Forward+. In *SIGGRAPH Asia 2012 Technical Briefs* (2012), pp. 18:1–18:4. 2

[Kaj86] KAJIYA J. T.: The Rendering Equation. *Computer Graphics (SIGGRAPH)* (1986), 143–150. URL: http://doi.acm.org/10.1145/15922.15902, doi:10.1145/15922.15902. 2

[KWR*17] KELLER A., WÄCHTER C., RAAB M., SEIBERT D., VAN ANTWERPEN D., KORNDÖRFER J., KETTNER L.: The Iray Light Transport Simulation and Rendering System. arXiv, https://arxiv.org/abs/1705.01263, 2017. 1, 2

[LYMT06] LAUTERBACH C., YOON S. E., MANOCHA D., TUFT D.: RT-DEFORM: Interactive ray tracing of dynamic scenes using BVHs. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing* (2006), pp. 39–46. 2, 3

[MC19] MOREAU P., CLARBERG P.: Importance sampling of many lights on the GPU. In *Ray Tracing Gems*, Haines E., Akenine-Möller T., (Eds.). Apress, 2019, pp. 255–283. http://raytracinggems.com. 2, 3, 4

[OA11] OLSSON O., ASSARSSON U.: Tiled Shading. *Journal of Graphics, GPU, and Game Tools 15*, 4 (2011), 235–251. 2

[OC17] O'DONNELL Y., CHAJDAS M. G.: Tiled Light Trees. In *Symposium on Interactive 3D Graphics and Games* (2017), pp. 1:1–1:7. 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics 29*, 4 (July 2010), 66:1–66:13. URL: http://doi.acm.org/10.1145/1778765.1778803. 3

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, third ed. Morgan Kaufmann, 2016. 2, 4

[PPD98] PAQUETTE E., POULIN P., DRETTAKIS G.: A Light Hierarchy for Fast Rendering of Scenes with Many Lights. *Computer Graphics Forum 17* (1998), 63–74. 2

[SKW*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. In *Proceedings of High-Performance Graphics* (2017), pp. 2:1–2:12. 4, 6

[SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient Estimation for Real-Time Adaptive Temporal Filtering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 2 (2018), 24:1–24:16. 5

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics 15*, 1 (1996), 1–36. 2

[TH16] TOKUYOSHI Y., HARADA T.: Stochastic Light Culling. *Journal of Computer Graphics Techniques 5*, 1 (2016), 35–60. 2

[vdB97] VAN DEN BERGEN G.: Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools 2*, 4 (1997), 1–13. URL: https://doi.org/10.1080/10867651.1997.10487480. 2

[VG95] VEACH E., GUIBAS L. J.: Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of SIGGRAPH* (1995), pp. 419–428. 4

[VKK18] VÉVODA P., KONDAPANENI I., KŘIVÁNEK J.: Bayesian online regression for adaptive direct illumination sampling. In *ACM Transactions on Graphics* (2018), pp. 125:1–125:12. 2

[War91] WARD G. J.: Adaptive Shadow Testing for Ray Tracing. In *Eurographics Workshop on Rendering* (1991), pp. 11–20. 2

[WBS03] WALD I., BENTHIN C., SLUSALLEK P.: Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), pp. 74–81. URL: http://dl.acm.org/citation.cfm?id=882404.882415. 2

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray Tracing Deformable Scenes Using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics 26*, 1 (Jan. 2007). URL: http://doi.acm.org/10.1145/1189762.1206075. 2, 3

[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 24*, 3 (2005), 1098–1107. 2

[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 (July 2014), 143:1–143:8. URL: http://doi.acm.org/10.1145/2601097.2601199. 3

[ZS95] ZIMMERMAN K., SHIRLEY P.: A Two-Pass Solution to the Rendering Equation with a Source Visibility Preprocess. In *Rendering Techniques* (1995), pp. 284–295. 2