Adaptive Environment Sampling on CPU and GPU

Asen Atanasov
Chaos Group

Vladimir Koylazov
Chaos Group

Blagovest Taskov
Chaos Group

Alexander Soklev
Chaos Group

Vassillen Chizhov
IMPRS, Saarland University

Jaroslav Křivánek
Render Legion
Charles University, Prague

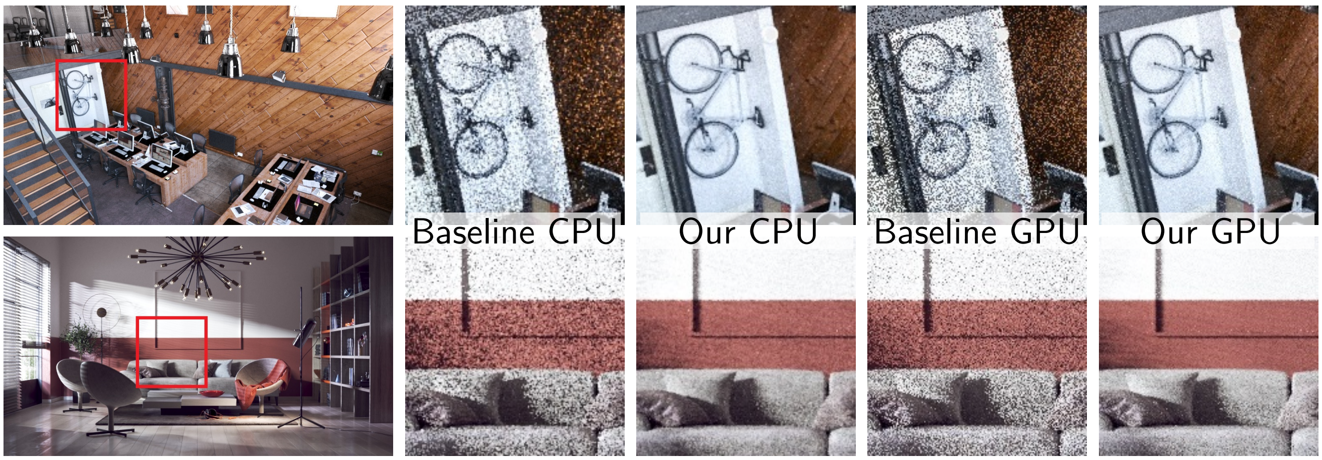

Figure 1: "Office" and "Living room" scenes rendered with classical environment sampling (*Baseline*) and our adaptive strategy. We present both CPU and GPU implementation results and show that our algorithm produces much cleaner images in the same time. The effective speedup, measured as the time to achieve the same noise level, for CPU/GPU implementations is, respectively: "Office" - 6.6/3.8 and "Living room" - 2.7/2.4. "Office" scene courtesy of Evermotion.

ABSTRACT

We present a production-ready approach for efficient environment light sampling which takes visibility into account. During a brief learning phase we cache visibility information in the camera space. The cache is then used to adapt the environment sampling strategy during the final rendering. Unlike existing approaches that account for visibility, our algorithm uses a small amount of memory, provides a lightweight sampling procedure that benefits even unoccluded scenes and, importantly, requires no additional artist care, such as manual setting of portals or other scene-specific adjustments. The technique is unbiased, simple to implement and integrate into a render engine. Its modest memory requirements and simplicity enable efficient CPU and GPU implementations that significantly improve the render times, especially in complex production scenes.

SIGGRAPH '18 Talks, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5820-0/18/08.
https://doi.org/10.1145/3214745.3214808

CCS CONCEPTS

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

KEYWORDS

image-based lighting, visibility caching, importance sampling

ACM Reference Format:
Asen Atanasov, Vladimir Koylazov, Blagovest Taskov, Alexander Soklev, Vassillen Chizhov, and Jaroslav Křivánek. 2018. Adaptive Environment Sampling on CPU and GPU. In *Proceedings of SIGGRAPH '18 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214808

1 INTRODUCTION

Image-based lighting (IBL) is an irreplaceable tool in production rendering used to light a scene with an environment map. We are interested in computing the reflected radiance L_r due to distant environment illumination L_d using the reflection equation $L_r(\mathbf{x}, \mathbf{o}) = \int_{\mathcal{H}^2(\mathbf{n})} f_r(\mathbf{x}, \mathbf{o}, \mathbf{i}) L_d(\mathbf{i}) V(\mathbf{x}, \mathbf{i})(\mathbf{n} \cdot \mathbf{i}) d\mathbf{i}$, where the integration is over the hemisphere $\mathcal{H}^2$ defined by the normal $\mathbf{n}$ at the shading point $\mathbf{x}$, $\mathbf{i}$ and $\mathbf{o}$ are incident and outgoing light directions, f_r is the BRDF, and V is the environment visibility. IBL is often a major source of noise, especially in interiors, because of occlusion. This has been traditionally addressed by placing portals and recently we have seen substantial improvements in this direction [Bitterli et al. 2015]. However, portals are not an effective approach:

the user is forced to spend time setting them up and has to understand how to use them effectively. This negatively affects the usability of the renderer [Karlík 2014]. For this reason, we seek a solution that can handle visibility in a way that is entirely transparent to the user. Bitterli et al. [2015] provide an overview of existing techniques and point out that those addressing visibility generally suffer from high memory usage, expensive computation or complex data structures, which limits their practicality, especially when used on GPUs. Furthermore, in favour of the SIMD architecture, we prefer grid-based look up over complex data structure traversal, because the latter results in higher code divergence [Bialas and Strzelecki 2015]. Lastly, a solution that improves occluded scenes but does not slow down unoccluded scenes is highly desirable.

2 OUR ALGORITHM

We divide the environment map into $T_u \times T_v$ equal-sized tiles and introduce a $G_x \times G_y$ spherical grid in the camera space which we call the *light grid*. Our algorithm operates in two phases: learning and rendering. The learning phase estimates and stores the contribution of each environment tile to each light grid cell. In the rendering phase we use the stored information to importance sample the environment map tiles based on their precomputed contribution [Cline et al. 2008]. A direction inside the chosen tile is then sampled proportionally to the environment intensity.

2.1 Learning phase

The light grid consists of $G = G_x G_y$ cells, each holding an array with $T = T_u T_v$ values, representing the contributions of the map tiles to the cell. A fixed number of camera paths are traced in the scene and for each path vertex – whether due to a primary ray or a secondary, GI ray – we determine its corresponding grid cell by projecting the vertex back to the camera center. We then importance sample the environment map and determine the sample's contribution, which is accumulated in the corresponding grid cell/map tile entry. This works well even when multiple or semi-transparent objects project to the same cell (see the supplementary document). At the end of the learning phase each cell c_i has T values $v_{ij}, j = 1 \ldots T$, that approximate the average radiance reaching the cell due to illumination from environment map tiles. Finally, for each cell these values are normalized and a CDF is constructed to facilitate sampling [Pharr et al. 2016].

Our default rendering pipeline starts with a brief irradiance caching phase. We take advantage of the samples generated in this prepass to calculate the tile sampling distributions. In this way the learning phase introduces a negligible overhead to the whole rendering process.

2.2 Rendering phase

During the actual rendering we combine one BRDF sample and one sample from our modified environment sampling strategy using MIS [Veach 1998]. For a given shading point, we look up its light grid cell and use its CDF to pick an environment map tile t_j, see Figure 2. The probability of picking this tile is p_{ij}. Then we generate an incident light direction $\mathbf{i}$ over this tile with probability $p_{t_j} \propto \frac{L_d(\mathbf{i})}{L_d(t_j)}$, where $L_d(\mathbf{i})$ is the environment map radiance along $\mathbf{i}$, $L_d(t_j)$ is the average unoccluded radiance due to illumination from tile t_j. As a result, our modified environment sampling probability is $p = p_{ij} p_{t_j}$. Note that our adaptive environment sampling is unbiased. It is possible that some tiles that are visible, but dark and partially occluded, have probabilities p_{ij} equal to zero. These cases do not cause any issues in practice since the BRDF sampling strategy is efficiently sampling these tiles.

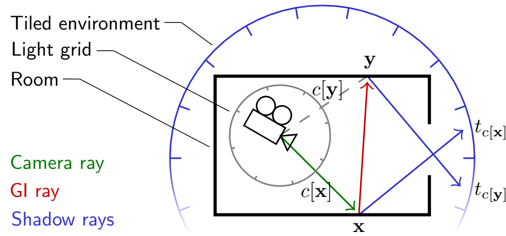

Figure 2: Rendering phase: Shading points x and y are back-projected to the camera to determine their respective light grid cells, $c[\mathbf{x}]$ and $c[\mathbf{y}]$. A visible environment tile, $t_{c[\mathbf{x}]}$ and $t_{c[\mathbf{y}]}$ respectively, is then chosen based on the cell CDF. Finally, shadow rays (blue) are sampled over the selected tiles.

2.3 Implementation and results

The algorithm is implemented in two separate production rendering engines (one CPU and one GPU). Tests on user production scenes show between 10% and 700% speedup. Two example interiors are shown in Figure 1. We observed that light grid resolution $(G_x, G_y) = (100, 50)$ and the environment map tiling $(T_u, T_v) = (16, 32)$ work well for production scenes and result in a modest 10MB memory usage. Coarser or finer subdivision gradually reduce the performance. We trace 10^6 camera paths in the learning phase which usually takes less than 1% of the total render time. The learning phase accumulation is implemented using fetch-and-add instructions, avoiding synchronization locks.

The algorithm requires a data structure that can efficiently draw samples, proportional to the map intensity, from the whole environment map as well as from map tiles individually. For this purpose we use an image sampler, based on a summed area table (SAT), that can draw samples from arbitrary rectangular subregions [Bitterli et al. 2015]. Single-precision SAT is notorious for its rounding error and fails to sample accurately large HDR images. Therefore, we build it in double precision and remap it to the 32-bit integer-value SAT. In the context of sampling, we found that the integer-valued SAT outperforms its single-precision counterpart in every way. Details are provided in the supplementary document.

REFERENCES

Piotr Bialas and Adam Strzelecki. 2015. Benchmarking the cost of thread divergence in CUDA. *CoRR* abs/1504.01650 (2015). arXiv:1504.01650 http://arxiv.org/abs/1504.01650

Benedikt Bitterli, Jan Novák, and Wojciech Jarosz. 2015. Portal-Masked Environment Map Sampling. *Computer Graphics Forum (Proc. of EGSR)* 34, 4 (June 2015).

David Cline, Daniel Adams, and Parris Egbert. 2008. Table-driven Adaptive Importance Sampling. In *Proceedings of the Nineteenth Eurographics Conference on Rendering (EGSR '08)*. Eurographics Association, 1115–1123.

Ondřej Karlík. 2014. It's all about usability. In *ACM SIGGRAPH 2014 Courses (SIGGRAPH '14)*. ACM, New York, NY, USA, Article 17, 6 pages.

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

Eric Veach. 1998. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph.D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.