Ambient Light Caching via Approximate Photon Mapping

Pavlos Makridis
Supervisor(s): Mark van de Ruit, Elmar Eisemann
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Ambient Light Caching via Approximate Photon Mapping

P. Makridis

Supervisor: M. van de Ruit

Responsible Professor: E. Eisemann

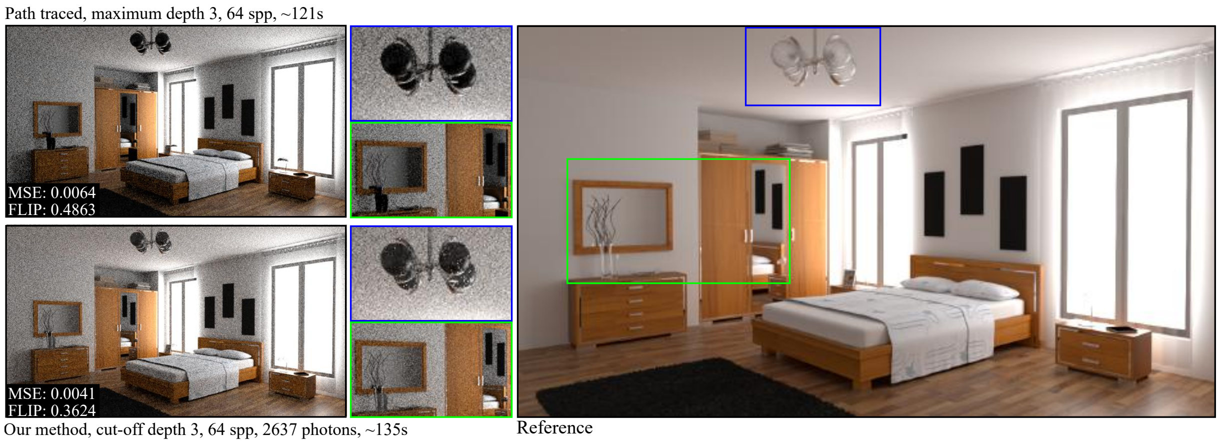

Path traced, maximum depth 3, 64 spp, ~121s
MSE: 0.0064
FLIP: 0.4863
MSE: 0.0041
FLIP: 0.3624
Our method, cut-off depth 3, 64 spp, 2637 photons, ~135s
Reference

Figure 1: *Our method improves the convergence of areas with challenging indirect illumination paths by using a photon-based cache after a specified path length has been reached.*

Abstract

Indirect illumination is an essential part of realistic computer-generated imagery. However, accurate calculation of indirect illumination comes at high compute costs. To this end, we replace lengthy indirect illumination paths by employing an ambient light cache based on photon mapping principles. By only performing cache queries after a certain path length has been reached, we show that significantly fewer photons than traditional photon mapping techniques are sufficient and that an inaccurate data structure can be used to store them. Despite these simplifications, our method generally outperforms unidirectional path tracing while adding little time overhead.

1. Introduction

Physically-based rendering is the golden standard for modern computer-generated imagery, as it allows for creating photorealistic images. This property is essential for various fields. For example, it is extensively used in the movie industry to create immersive experiences for the audience and in architecture to help professionals visualise their designs.

Typically, photorealistic rendering techniques rely on light transport algorithms. These algorithms estimate the colour of each pixel in an image by constructing paths connecting it to a light source in the scene. Those paths are created by casting rays from the camera and following how those rays scatter through the scene. To carry out this calculation, the rays that leave from the camera and the rays that result from scattering on the scene's surfaces need to be tested for intersections with the scene's geometry. Moreover, to achieve acceptable noise levels in the output, this process must be repeated multiple times for every pixel.

The above light transport algorithms allow for simulating various everyday light phenomena. Out of these phenomena, this work focuses on indirect illumination. Indirect illumination is the incident light on a surface from other reflective surfaces [MS22]. A typical example of a scene with indirect illumination is a room lit from the outside like the one in Figure 2.

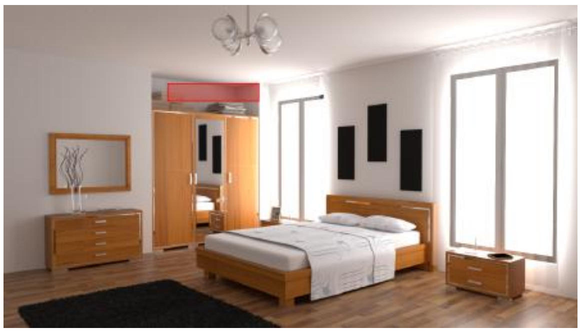


Figure 2: ***Example of indirect illumination****. A typical example of a scene with indirect illumination. The red square highlights an area for which the indirect illumination calculation is particularly costly.*

Unfortunately, calculating indirect lighting comes at a high computational cost. This cost arises because indirect light paths are usually lengthy, implying that many intersection tests must be performed. Consider how, in Figure 2, a camera ray that reaches the highlighted area would propagate towards the light. This increase in length, combined with the high sample count per pixel, causes the rendering cost to proliferate.

This work aims to minimise the indirect lighting calculation cost. This reduction is achieved through a novel application of the widely employed photon mapping technique [JC95]. Concretely we use a simplified version of photon mapping to create a flux cache, to which lookups are performed once a specified path depth has been reached. We showcase that our method relies on significantly fewer photons than traditional photon mapping and that a simple spatial hash grid structure can be used to store them. Ultimately, we arrive at a biased technique that generally outperforms unbiased unidirectional path tracers.

Although there is an expansive body of work investigating precomputed ambient terms [WRC88, KGPB05, MRNK21], our contribution remains novel by accurately calculating the first few scatterings of each light path. Consequently, we present approximations that make our cache simple and efficient to construct. Thereby, we achieve lower overhead than contemporary techniques.

In the rest of this paper, we first cover the necessary background and related work (Section 2), after which we present the exact working of our technique (Section 3) and briefly discuss some of the implementation details (Section 4). Then, we present an evaluation of our cache (Section 5) and reflect on the responsible research aspects of this work (Section 6). Lastly, we give discuss various aspects of our method (Section 7) before concluding (Section 8).

2. Background and Related Work

Radiometry To describe light phenomena, rendering algorithms typically rely on radiometric tools [PJH16]. To this end, it is worth reviewing the three fundamental radiometric quantities we use in this work. Those are *flux*, *exitant radiance* and radiance. *Flux* or *power* is the total amount of energy passing through a surface or region of space per unit time. *Exitant radiance* is the the area density of flux leaving a surface. Lastly, *radiance* is flux density per unit area, per unit solid angle. Our method relies on precomputing and caching flux on certain areas and then converting it to exitant radiance during rendering.

The rendering equation Physically accurate light transport algorithms evaluate the *rendering equation* [Kaj86] which is given as:

$$L(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega f(x,\omega_i,\omega_o)L(x,\omega_i)\cos(\theta_i)d\omega_i \tag{1}$$

Here $L(x,\omega_o)$ is the exitant radiance from point x to direction ω_o, and it consists of the sum of two terms. The first term is the emitted radiance at x. The second term expresses the radiance coming to x, from all directions ω_i of the hemisphere Ω, that gets scattered back towards ω_o. The function f is the *bidirectional scattering distribution function* (BSDF) which describes the way the surface at x scatters light from direction ω_i to ω_o [Vea98]. In this paper, we focus on the second term, referred to as:

$$L_i(x,\omega_o) = \int_\Omega f(x,\omega_i,\omega_o)L(x,\omega_i)\cos(\theta_i)d\omega_i \tag{2}$$

Generally, Equation 2 does not have an analytical solution. Because of this, rendering techniques rely on numerical approximations of it. Specifically, Monte Carlo integration is well suited for this purpose [Laf96]. A Monte Carlo estimator for it is:

$$L_i(x,\omega_o) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x,\omega_i,\omega_o)}{p(\omega_i;\omega_o)}L(x,\omega_i)\cos(\theta_i), \tag{3}$$

where N is the number of samples taken at each point and $p(\omega_i;\omega_o)$ is the probability density function describing the probability of sampling direction ω_i when coming from direction ω_o.

Cache-based techniques Cache-based rendering techniques leverage previous results or intermediate computations to accelerate the evaluation of Equation 2. The two most relevant cache-based techniques for this work are irradiance caching and photon mapping. We present those techniques in the following paragraphs.

Irradiance caching [WRC88, War94], works on the observation that indirect illumination varies smoothly over surfaces. Based on this observation, irradiace caching methods fully evaluate indirect illumination only on select points and then interpolate it for the rest. While effective, this strategy introduces several non-trivial challenges, such as picking caching points, developing effective interpolation techniques and handling non-diffuse surfaces. Although several improvements have been proposed [WH92, KGPB05], they make the construction of irradiance caches rather complicated.

Another technique that we classify as cache-based is *photon mapping* [JC95]. Photon mapping is a biased but consistent two-pass approach that builds on bidirectional rendering techniques. In bidirectional techniques, paths are constructed both from the camera to light sources and from light sources to the camera [VG95]. On the first pass, rays are traced from the light to the scene. These rays act like those used in regular path tracing, with the difference that in all diffuse intersection points, they place a *photon* which contains flux and direction information. On the second pass, rays are cast from the camera to the scene. When those rays intersect the scene, all nearby photons are accumulated to produce an irradiance value.

Like the irradiance cache, photon maps come with certain drawbacks. Notably, photon mapping typically requires large amounts of photons. Those photons need to be traced and stored, introducing time and memory overheads. Methods to reduce this memory and render time overhead have been proposed [SJ09], but they slow down the algorithm's preprocessing phase.

3. Method

We now present our method. First, we summarise our initial approaches and highlight their shortcomings. Then, we give an overview of our technique, after which we provide a reformulation of the rendering equation that incorporates our ambient light cache.

Early attempts Initially, several straightforward strategies for the ambient cache were tested. First, we experimented with precomputing one global ambient term and with sampling lights without accounting for visibility. These approaches did not account for the scene's geometry and thus caused light leaking. To improve that, we tested an approach that used photon mapping to fully precompute the exitant radiance in the centroids of the grid's voxels. This approach proved promising but was particularly sensitive to parameter changes. With these observations in mind, we present an improved method below.

Algorithm overview Our method acts in the following way. First, it receives the scene as input and builds a spatial hash voxel grid around it. Then it traces light paths from the scene's light sources, computing photons at the diffuse intersection points and caching them in the grid's voxels. At this point, the preprocessing step has been completed, and rendering can proceed normally. During rendering, light scatterings are accurately calculated until the paths reach a defined *cut-off depth*. Because photons are only stored on diffuse surfaces, if the last intersected surface is specular or transmissive, the path is allowed to proceed until a diffuse surface is found. Then, the last intersection point is given as input to the cache. The cache places the point in the corresponding voxel of the grid, and the photons of that voxel are used to produce an exitant radiance estimate. An explanation of this estimation is provided in the following paragraphs.

The cache equation We now introduce the cache equation, which approximates Equation 2. We rewrite it to consider a depth factor d, which is the path depth at which the computation occurs ($d = 0$ being the first intersection point). We also define the cache lookup result as the function $C(x_d)$ that is based on a point x at depth d. As the cache is used once the cut-off depth c has been reached, the second term of the rendering equation is split into two cases. For the case of $d < c$, it remains:

$$L_{o,d}(x_d, \omega_{r,d}) = \int_{\Omega} f(x, \omega_{i,d}, \omega_{r,d}) L(x_{d+1}, \omega_{i,d}) \cos(\theta_{i,d}) d\omega_{i,d}$$

whereas for the case $d = c$ it becomes:

$$L_{o,d}(x_d, \omega_{r,d}) \approx C(x_d)$$

In the following, we elaborate on the $d = c$ case. To derive an irradiance estimate based on the nearby photons, we use Shirley et al.'s [SWH*95] particle-based irradiance function:

$$C(x_d) = \frac{1}{Nh^2} \sum_{i=1}^{N} k\left(\frac{||x_d - p_i||}{h}\right) \phi_i$$

This equation describes the sum of the contribution of N photons at a point x_d. The p_i and ϕ_i terms are the position and the weight of the i-th photon respectively, while $k(\frac{||x_d - p_i||}{h})$ is the scale factor of the photon's weight, which is determined by kernel density estimation. In our case, we use the Silverman kernel [SWH*95]:

$$k(x) = \frac{3}{\pi} max(0, (1 - x^2)^2)$$

By using a voxel grid, finding all the photons for which $k(\frac{||x_d - p_i||}{h})$ is positive requires constant time since we know that all photons in a voxel are within a certain distance from the intersection point. Furthermore, contrary to typical photon mapping techniques, our method does not need to store incident direction information for each photon.

4. Implementation

We implement our technique in the C++ physically-based renderer PBRT-v3 [PJH16]. Concretely we adjust the *path integrator* to include the cache creation preprocessing step. This step traces light rays from the scene's light sources and stores photons at the intersection points with the world's geometry. Note that the light ray propagation is terminated via either Russian roulette [AK90] or when a maximum depth has been reached.

When implementing the photon accumulation step, some details must be kept in mind. First, to minimise light leakage, photons are only used if they lie on the same plane as the point for which the estimation occurs. Moreover, since KDE is performed in voxels of width w, the kernel width that must be used is $\sqrt{2}w$, since this is the maximum distance two points in the same voxel can have. Last, note that performance can increase by turning divisions by a constant into multiplications of the constant's inverse and only applying them after calculating the sum of individual photon contributions.

5. Evaluation

We evaluate our technique in scenes with significant indirect illumination. First, we investigate the impact of our methods' parameters. Then, with the intuition gained by the parameter analysis, we derive a configuration with which we evaluate our method in two different scenes. For each scene, we compare the performance of our method at different cut-off depths with the performance of an unbiased unilateral renderer with the maximum depth set at the cache's cut-off depth. Last, to judge the quality of the renders, we rely on the mean squared error (MSE) and the recently introduced FLIP metric [ANAM*20]. Since our method is biased, we rely on FLIP to showcase the perceived difference between images.

Timing and system Throughout the evaluation, we compare the speed of our technique with that of an unbiased unidirectional path renderer. Those timings are performed in a Dell G3 laptop running Windows 10 on a 12th core Intel i7 8th generation CPU. We use the *hyperion* benchmarking tool with three warm-up runs and five repetitions to time the rendering process. Moreover, to time the preprocessing step, we rely on profiling the specific part of the code and averaging the results. Note that there still is some noise in the timings due to the system's instabilities, but it does not affect the general trend of the results.

Parameter evaluation Our method contains 3 parameters. The voxel width, the maximum photon tracing depth and the number of photons traced (photon count). Note that the number of photons stored is larger than the photons traced (because a photon ray can intersect a scene multiple times). We define a baseline configuration with a voxel width of 1, a photon tracing depth of 5 and a photon count of 500. To investigate the effect of each, we fix the other two and evaluate the speed and quality of our technique. These evaluations are performed on the Veach-Bidir scene [Bit16]. Figure 5 showcases how our baseline compares with a reference image and a path traced image with a maximum depth of 3, while Figure 4, showcases the results of the parameter evaluation and the times it took to produce them.

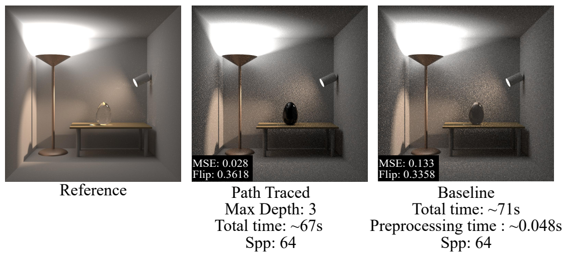


Figure 3: ***Parameter evaluation baselines.*** *We showcase the results of our technique next to those of a path tracer, with the cut-off and maximum depth, respectively is set to 3. Both of these renders are produced with 64 samples per pixel.*

We see that the error metrics show minor variations across parameter changes. In Figure 5, we see that the radius plays a significant role in the appearance of the glass egg. Moreover, here we also observe some artifacts due to the cache's grid. Furthermore, we note that the time required for the preprocessing step of our method is in the order of microseconds and is affected only by the number of photons traced.

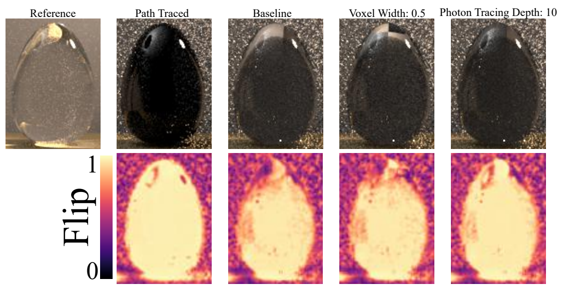


Figure 5: ***Closer inspection of parameter impact.*** *We see that the radius impacts the appearance of the glass egg on the scene the most by increasing the visible artifacts.*

To derive the configuration used in the method evaluation, we combine the parameter values that gave the best results. Therefore, we set the voxel width to 1, the photon count to 1000 and the photon tracing depth to 3. Note that this approach does not guarantee the optimality of results in the evaluation scenes; however, as the alternative implies overfitting, we opt for those more general values.

Method evaluation We evaluate our method against the Veach-Ajar, and Bedroom scenes [Bit16]. To evaluate the performance of our method, we vary the cut-off depth and compare the results with a path-traced render of the same maximum depth. Our results are showcased in Figure 6, while Figure 5 showcases the impact that varying the maximum depths has on the error metrics. In addition Figures 8

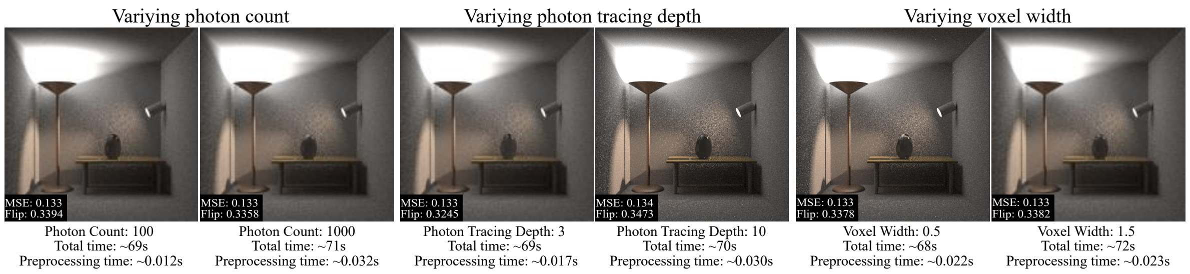

Variying photon count
Variying photon tracing depth
Variying voxel width
MSE: 0.133 Flip: 0.3394
Photon Count: 100 Total time: ~69s Preprocessing time: ~0.012s
MSE: 0.133 Flip: 0.3358
Photon Count: 1000 Total time: ~71s Preprocessing time: ~0.032s
MSE: 0.133 Flip: 0.3245
Photon Tracing Depth: 3 Total time: ~69s Preprocessing time: ~0.017s
MSE: 0.134 Flip: 0.3473
Photon Tracing Depth: 10 Total time: ~70s Preprocessing time: ~0.030s
MSE: 0.133 Flip: 0.3378
Voxel Width: 0.5 Total time: ~68s Preprocessing time: ~0.022s
MSE: 0.133 Flip: 0.3382
Voxel Width: 1.5 Total time: ~72s Preprocessing time: ~0.023s

Figure 4: ***Parameter Evaluation.*** *We evaluate our method's parameters. For every pair, we showcase the impact on the MSE and Flip metrics, how long the image took to produce and how much time was spent on the preprocessing step.*

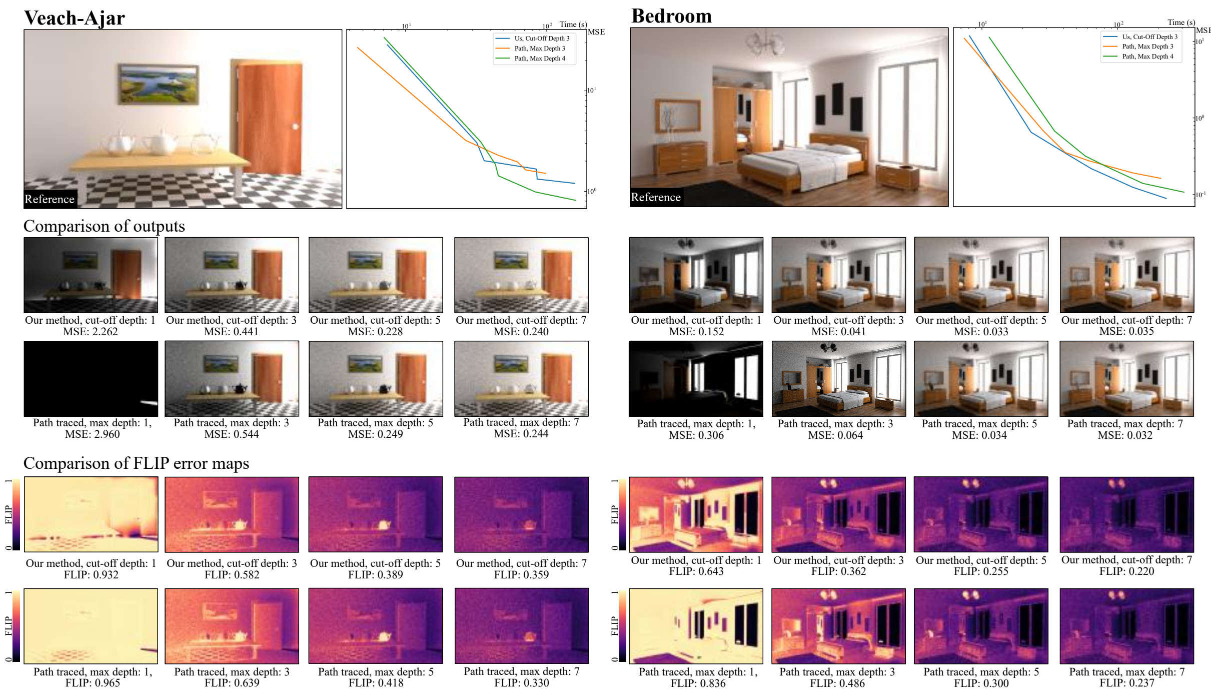

Veach-Ajar
Bedroom
Reference
Time (s)
MSE
Us, Cut-Off Depth 3
Path, Max Depth 3
Path, Max Depth 4
Comparison of outputs
Our method, cut-off depth: 1 MSE: 2.262
Our method, cut-off depth: 3 MSE: 0.441
Our method, cut-off depth: 5 MSE: 0.228
Our method, cut-off depth: 7 MSE: 0.240
Path traced, max depth: 1, MSE: 2.960
Path traced, max depth: 3 MSE: 0.544
Path traced, max depth: 5 MSE: 0.249
Path traced, max depth: 7 MSE: 0.244
Our method, cut-off depth: 1 MSE: 0.152
Our method, cut-off depth: 3 MSE: 0.041
Our method, cut-off depth: 5 MSE: 0.033
Our method, cut-off depth: 7 MSE: 0.035
Path traced, max depth: 1, MSE: 0.306
Path traced, max depth: 3 MSE: 0.064
Path traced, max depth: 5 MSE: 0.034
Path traced, max depth: 7 MSE: 0.032
Comparison of FLIP error maps
FLIP
Our method, cut-off depth: 1 FLIP: 0.932
Our method, cut-off depth: 3 FLIP: 0.582
Our method, cut-off depth: 5 FLIP: 0.389
Our method, cut-off depth: 7 FLIP: 0.359
Path traced, max depth: 1, FLIP: 0.965
Path traced, max depth: 3 FLIP: 0.639
Path traced, max depth: 5 FLIP: 0.418
Path traced, max depth: 7 FLIP: 0.330
Our method, cut-off depth: 1 FLIP: 0.643
Our method, cut-off depth: 3 FLIP: 0.362
Our method, cut-off depth: 5 FLIP: 0.255
Our method, cut-off depth: 7 FLIP: 0.220
Path traced, max depth: 1, FLIP: 0.836
Path traced, max depth: 3 FLIP: 0.486
Path traced, max depth: 5 FLIP: 0.300
Path traced, max depth: 7 FLIP: 0.237

Figure 6: ***Method evaluation.*** *We showcase the performance of our method in the Veach-Ajar and Bedroom scenes. We compare our method with an unbiased unilateral path tracer across different cut-off and maximum depths. We display the corresponding renders with their MSE and the FLIP error maps. For select depths, we plot the convergence behaviour of our method against that of a path tracer with the same maximum depth and that of a path tracer that is allowed one additional bounce.*

and 9 display the values the cache returns at every point. Our method generally delivers lower error rates and is most effective on shorter path lengths. This behaviour is expected since increasing the path after a certain point for each scene makes little difference in the output. Moreover, observe that although our method adds delays during rendering, it is generally faster and better in quality than the result of incrementing the path depth. This advantage is showcased strongly in the convergence graph for the Bedroom scene shown on the top left of Figure 6. In contrast, for the Veach-Ajar scene, we see that the benefits of our technique are not that strong since a lower maximum depth is required and intersection tests are fast.

Inspecting the cache values showcased in Figures 8 and 9, we see that they are generally low and very discretely split because of the grid. Nonetheless, as our renders show, this does not create artifacts when the maximum depth is larger than 1. This effect occurs even in the Veach-Ajar scene, where the photon map is primarily black due to the low photon tracing depth we have used, yet the method still performs adequately. Overall our method indicates that even

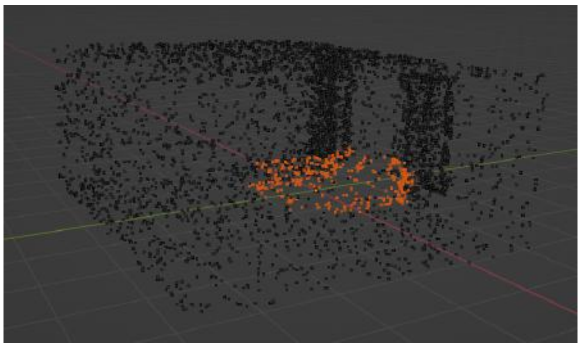


Figure 7: ***Bedroom photon spread.*** *The spread of photons in the bedroom scene visualized as a point cloud in Blender. The highlighted part indicates the location of the bed.*

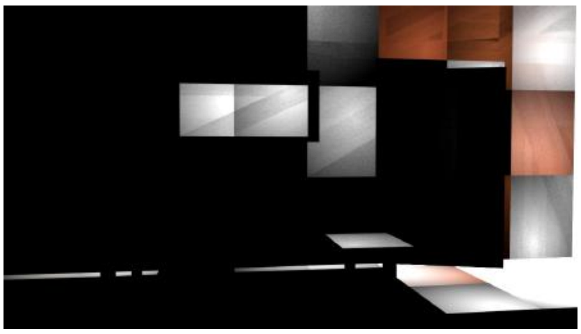


Figure 8: ***Veach-Ajar photon map.*** *The cache outputs for the Veach-Ajar scene directly visualized.*

inaccurate estimations can improve the perceived quality of the outcome regarding indirect illumination.

Simultaneously, on the photon map of the bedroom scene (Figure 9), notice that the bed and the carpets appear entirely dark. However, if we visualise the photon distribution (shown in Figure 7), we see that photons are stored there. Those photons are not used because our method minimises light leaking by only accounting for photons on the same plane as the estimation point. In this case, the geometry of those parts, combined with the low photon count, causes the check we perform to fail. Another failure case of this check can be seen in Figure 8 where leaks through the back wall. This leakage occurs because the floor extends behind the wall photons from the light are deposited there. When the cache estimates the irradiance for a point on the floor in front of the wall, these photons are still accounted for because they are located on the same plane.

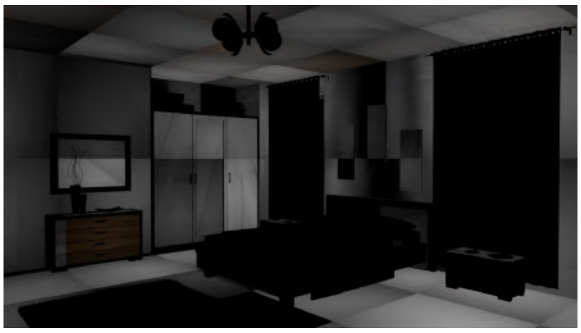
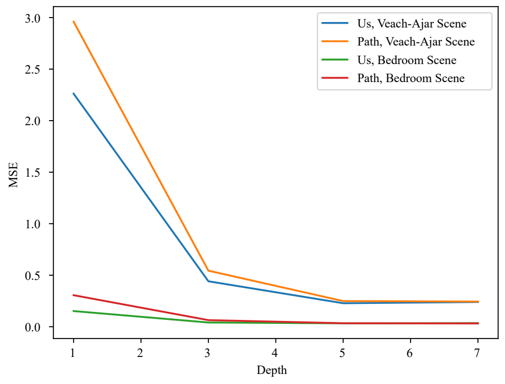


Figure 9: ***Bedroom photon map.*** *The cache outputs for the Bedroom scene directly visualized.*



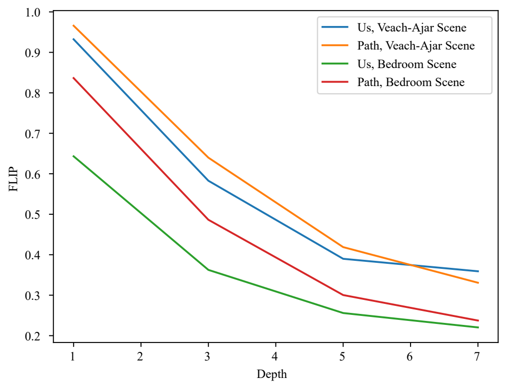


Figure 10: ***Error metrics across depths.*** *We compare the MSE and FLIP metrics across different maximum path depths between our method and a unbiased unilateral path-tracer. We show that our method decreases errors quicker than the path integrator.*

6. Responsible Research

We now present how responsible research practices are applied in this work. In the first paragraph, we explain how our research is reproducible. Then, we showcase how the principles of the *Netherlands Code of Conduct for Research Integrity* are followed in this work.

Reproducibility in computer graphics is essential because it allows for verifying and improving existing techniques and enables them to be used for comparisons with future methods. To ensure the reproducibility of our method, we include implementation details in Section 4. Moreover, the method is implemented on the widely-known and open-source PBRT-v3 renderer [PJH16]. The renderer is straightforward to install and use, thus addressing the replicability of build process concerns raised by Bonneel and collaborators [BCDM20]. Last, we conduct our method's evaluation on publicly available scenes so that our claims and exact results can be verified. In this way, we hope that our method can be effortlessly reproduced.

This work adheres to the principles of research integrity showcased in the *Netherlands Code of Conduct for Research Integrity* [KNN*18]. To strive for *honesty*, we underline our methods' shortcomings in Section 7. We aim for *scrupulousness* by using best practices for timing our technique, such as warm-up and multiple runs. To ensure transparency, Section 5 provides a detailed description of how our evaluation data is obtained. *Independence* is achieved by impartially evaluating our results. Last, we adhere to *responsibility* since the field of physically-based rendering applies to various fields, making this work societally relevant. By following the above, we aim to ensure the scientific integrity of this work.

7. Discussion

We have shown that our method generally increases the quality of low maximum depth renders. Our technique introduces a preprocessing step in the order of microseconds, requiring significantly fewer photons than other photon mapping techniques. For comparison, Jensen's original photon mapping technique and its progressive variants typically require more than 100K photons [JC95, HOJ08, HJ09], while later techniques reduce this to 10K photons [SJ09]. In contrast, we have shown that our technique operates adequately with less than 3K photons. Moreover, contrary to traditional photon mapping, which stores photons on a kd-tree, we store them in a spatial hash grid. In most cases, using the grid does not create visible artifacts. As a last note, our method adds some time overhead during rendering, but it is generally worth the increase in quality.

By examining our results, we can find directions for improving our technique. First, in the Veach-Bidir scene, artifacts caused by the voxel grid can be seen on the egg. These artifacts are caused because the glass scatters rays in directions with high differences in photon densities. Using a more accurate structure for storing the photons, such as a kd-tree, would improve our technique in this aspect. However, care must be taken so that the photon retrieval step also accumulates the photons. Otherwise, slowdowns are introduced since the nearest photons need to be iterated twice. In addition, using a kd-tree could provide efficient and exact range queries, allowing us to use a shrinking kernel instead of a fixed size one. With a shrinking kernel, a progressive reformulation of our technique, like the one used in progressive photon mapping [HJ09], could be used to make our technique unbiased. Note that since kd-tree construction times are of logarithmic order [Ben75], the preprocessing time will increase but considering the low photon count of our technique, this increase will probably be negligible. Last, our light leakage fix is not a perfect solution. As seen in the bedroom scene, it causes our technique to underestimate the indirect illumination, whereas, in the Veach-Ajar scene, light leaks through the wall. As an improvement, a photon-based radiance estimation technique that accounts for the scene's geometry, like the one showcased by Hey and Purgathofer [HP02], can be used.

8. Conclusions

This work has proposed a biased method for ambient light caching using a novel application of photon mapping. Our method has demonstrated that coarse approximations of indirect illumination can be effective as replacements for challenging light paths. We have shown that our method relies on a severely reduced photon count than traditional photon mapping techniques and a simple, inaccurate data structure. Despite these apparent reductions in quality, our method generally outperforms unbiased unidirectional path tracing. While it introduces some time overhead in rendering, typically, our technique is faster and more effective than incrementing the path length, making it particularly useful for scenes where intersection tests are exceptionally costly. In conclusion, we hope our method contributes to reducing compute costs for producing photorealistic imagery while showcasing that even simple approximations of indirect illumination can positively impact quality.

9. Acknowledgments

The authors would like to thank Matt Pharr, Wenzel Jakob, and Greg Humphreys for their work on PBRT-v3 [PJH16]. Additionally, we would like to thank Benedikt Bitterli for making publically available scenes for the PBRT renderer [Bit16]. Last but not least, we would like to thank Jorge Romeu Huidobro, Mathanrajan Sundarrajan, Tan Dao and Piyush Deshmukh for their support throughout this work.

References

[AK90] Arvo J., Kirk D.: Particle transport and image synthesis. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), pp. 63–66. 4

[ANAM*20] Andersson P., Nilsson J., Akenine-Möller T., Oskarsson M., Åström K., Fairchild M. D.: Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech. 3*, 2 (2020), 15–1. 4

[BCDM20] Bonneel N., Coeurjolly D., Digne J., Mellado N.: Code replicability in computer graphics. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 93–1. 7

[Ben75] Bentley J. L.: Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9 (sep 1975), 509–517. URL: https://doi.org/10.1145/361002.361007, doi:10.1145/361002.361007. 7

[Bit16] Bitterli B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 4, 7

[HJ09] Hachisuka T., Jensen H. W.: Stochastic progressive photon mapping. In *ACM SIGGRAPH Asia 2009 papers*. 2009, pp. 1–8. 7

[HOJ08] Hachisuka T., Ogaki S., Jensen H. W.: Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers*. 2008, pp. 1–8. 7

[HP02] Hey H., Purgathofer W.: Advanced radiance estimation for photon map global illumination. *Computer Graphics Forum 21*, 3 (2002), 541–545. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00704, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00704, doi:https://doi.org/10.1111/1467-8659.00704. 7

[JC95] Jensen H. W., Christensen N. J.: Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics 19*, 2 (1995), 215–224. 2, 3, 7

[Kaj86] Kajiya J. T.: The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4 (aug 1986), 143–150. URL: https://doi.org/10.1145/15886.15902, doi:10.1145/15886.15902. 2

[KGPB05] Kriváanek J., Gautron P., Pattanaik S., Bouatouch K.: Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 550–561. 2, 3

[KNN*18] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, VSNU: Nederlandse gedragscode wetenschappelijke integriteit, 2018. URL: https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:110600, doi:10.17026/DANS-2CJ-NVWU. 7

[Laf96] Lafortune E.: Mathematical models and monte carlo algorithms for physically based rendering. *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven 20* (1996), 74–79. 2

[MRNK21] Müller T., Rousselle F., Novák J., Keller A.: Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021). 2

[MS22] Marschner S., Shirley P.: *Global Illumination*. CRC Press, Taylor amp; Francis Group, 2022. 2

[PJH16] Pharr M., Jakob W., Humphreys G.: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 2, 3, 7

[SJ09] Spencer B., Jones M. W.: Into the blue: Better caustics through photon relaxation. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 319–328. 3, 7

[SWH*95] Shirley P., Wade B., Hubbard P. M., Zareski D., Walter B., Greenberg D. P.: Global illumination via density-estimation. In *Eurographics Workshop on Rendering Techniques* (1995), Springer, pp. 219–230. 3

[Vea98] Veach E.: *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998. 2

[VG95] Veach E., Guibas L.: Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 1995, pp. 145–167. 3

[War94] Ward G. J.: The radiance lighting simulation and rendering system. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), pp. 459–472. 2

[WH92] Ward G. J., Heckbert P. S.: *Irradiance gradients*. Tech. rep., Lawrence Berkeley Lab., CA (United States); Ecole Polytechnique Federale . . . , 1992. 3

[WRC88] Ward G. J., Rubinstein F. M., Clear R. D.: A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), pp. 85–92. 2