Eurographics Symposium on Rendering - Industry Track (2019)
T. Boubekeur and P. Sen (Editors)

Implementing One-Click Caustics in Corona Renderer

Martin Šik[1] and Jaroslav Křivánek[1,2]

[1]Chaos Czech a.s., Czech Republic
[2]Charles University, Prague

a) No caustics (21 min) b) With caustics (38 min)

Figure 1: *Our caustics solver automatically renders directly and indirectly visible caustics without requiring the user to set any technical parameters. The image on the left is rendered using path tracing that allows next event estimation across the water surface in 21 minutes. The full caustics solution on the right is rendered in 38 minutes with no such approximations. Image courtesy 3Darcspace studio.*

Abstract

This paper describes the implementation of a fully automatic caustics rendering solution in Corona Renderer. The main requirement is that the technique be completely transparent to the user, should not need any parameter setting at all, and be fully integrated into the interactive and progressive rendering workflow. We base our approach on an efficient subset of the vertex connection and merging algorithm, specifically a multiple importance sampling combination of path tracing and photon mapping. We rely on Metropolis sampling to guide photon paths into the relevant parts of the scene. While these underlying ideas have appeared in existing research work, numerous previously unaddressed issues and edge cases arise when one applies these ideas in practice. These include unreliable convergence of the Metropolis sampler in scenes with many light sources of different sizes and intensities, the "caustic in a stadium" problem (i.e., efficient rendering of small caustics in extremely large scenes), etc. We present the solutions we have developed to address such issues, yielding what we call "one-click caustics rendering". User feedback suggests that our approach substantially improves usability over methods previously implemented in comercially available software, all requiring the user to set various technical parameters.

1. Introduction

Caustics are an important visual phenomenon contributing to the perceived realism of scenes involving reflective and refractive surfaces. They are an important effect in rendering for VFX, product design, automotive industry, and – importantly for our use case – architectural visualization. Numerous solutions for rendering caustics in a Monte Carlo rendering framework have been proposed in the past, notably photon mapping [Jen96, HOJ08, HJ09], bidirectional path tracing [VG95, PLW98], and algorithms that achieve improved robustness by combining both these approaches [VKŠ*14, GKDS12, HPJ12, KGH*14]. But such combined solutions usually come with significant implementation complexity and run-time overhead over the much simpler unidirectional path tracing. As a result, they are rarely adopted in production rendering [KCSG18, CFS*18, FHL*18, GIF*18, BAC*18, KKK*18].

To address these issues, Grittmann et al. [GPGSK18] have recently proposed the lightweight photon mapping algorithm with the aim

DOI: 10.2312/sr.20191221

to enable caustics rendering without compromising the efficiency of simple path tracing.

Our goal is to implement a caustics rendering solution that is completely transparent to the user, does not rely on any user-settable parameters, has minimum overhead, and is fully integrated into our interactive and progressive rendering workflow. The implementation is based on Grittmann et al.'s work, where path tracing is combined with *selective* photon tracing using multiple importance sampling. We, however, depart from the original work in several important ways and we additionally resolve the various edge cases encountered in production rendering. These include namely issues with the convergence of a Metropolis photon tracer [HJ11, GRŠ*17] in scenes with many light sources of very different sizes and intensities, the "caustic in a stadium" problem (i.e., efficient rendering of small caustics in extremely large scenes), Our solution yields what we call "one-click caustics rendering", substantially improving usability over methods previously implemented in commercially available rendering software. It is now being shipped as an integral part of Corona Renderer.

2. Related Work

Existing caustics rendering solutions can be categorized into four groups: Bidirectional methods, Metropolis light transport, Path guiding, and Manifold next event estimation.

Bidirectional methods. In addition to tracing paths from the camera, bidirectional methods also trace paths from the light sources, to form the so called *photon paths*. Various bidirectional methods then utilize these paths in different ways. For example, the bidirectional path tracing [VG95, PLW98] uses a) unidirectional path tracing technique that randomly hits the light sources; b) path tracing technique that connects a camera path to the light sources; c) bidirectional connection between a camera subpath and a photon subpath; and d) and direct connection of photon paths to the camera. Caustics can be handled by some of these techniques, for instance by the direct connection of photon paths to the camera.

Bidirectional methods have their issues, though. Apart from the conceptual and implementation complexity, probably the most important one is determining where to trace the photon paths so as to achieve efficient rendering. This issue is especially important in larger scenes, where – without additional care – only a small fractions of photon paths reach the region visible by the camera. This is one of the main reasons why most rendering systems rely primarily on path tracing, which traces paths from the camera, effectively sidestepping the issue.

Metropolis light transport (MLT). MLT [VG97, ŠK19] utilizes the so called Metropolis sampling algorithm. It randomly searches for a path with a substantial contribution and once it finds it, it will slightly modify (or "mutate") this path to create several different contributing paths. This way, once the algorithm manages to sample one caustic path, it can efficiently explore the entire caustic. Metropolis light transport, however, produces some disturbing visual artifacts, especially conspicuous as distracting flickering in animations [ŠOHK16].

Path guiding. Path guiding applies machine learning methods to learn from the previous samples where to trace the subsequent samples [Jen95, VKŠ*14, HEV*, MGN17, KKN*18, MMR*18]. Instead of selecting the path directions at random, a guided path tracer utilizes a continuously trained distribution to choose more samples in the important directions. Training of the guiding distributions is, however, often slow and imprecise, and path guiding will thus often fail to find all the caustics.

Manifold next event estimation. The manifold next event estimation method [HDF15] uses differential geometry to iteratively find a connection path between a light source and a given surface point through a refractive interface. This method can be used to efficiently render caustics due to light refraction, but it will not help when caustics are created by reflection.

In summary, none of the existing solutions comes without issues. But the bidirectional methods currently produce the best caustics and their issues are the easiest to resolve; for this reason we have chosen them as the basis of our work.

3. Our Approach

Requirements. Our goal is to present the Corona Renderer users with a caustics solver capable of efficiently resolving any type of caustics: reflective and refractive, seen both directly or indirectly. To ensure good usability, we want this solver to have a minimum number of parameters, ideally just a single checkbox that turns the solver on and off. While we realize that the solver will certainly generate some overhead over an ordinary path tracer, our goal is to keep the overhead at a minimum. Finally, to prevent the solver from being "in the way", it is necessary that it behaves as if turned off in scene regions without caustics.

Solution overview. We take a robust bidirectional method known as Vertex Connection and Merging (VCM) [GKDS12, HPJ12] as a basis of our implementation. VCM uses all the path sampling techniques from bidirectional path tracing and combines them with radiance estimates from photon mapping [Jen96, HOJ08, HJ09]: As a path is traced from the camera, photon map lookups are performed at each intersection point (i.e. camera path vertex). The photon mapping technique is essential for rendering indirectly visible caustics, such as the ones seen at the bottom of a swimming pool. The full VCM algorithm can handle all kinds of caustics well and provides a solid basis for our work.

However, baseline VCM suffers from some issues that are in a direct contradiction with the requirements listed above:

Computational overhead. Baseline VCM can be easily several times slower than path tracing. Not only does it need to trace paths from both the light sources and the camera, it also needs to compute weights for their combinations, and perform the photon lookups. Using VCM in scenes that are well handled by path tracing is an overkill and would not be accepted by the users.

Photon guiding. Without guiding the photon paths to the relevant parts of the scene, the algorithm may converge very slowly. This issue is especially serious in architectural interior rendering, where skylight often reaches the interior through relatively narrow windows.

Number of photon paths. VCM traces the same number of photon paths as camera paths, but this can be quite wasteful for scenes featuring just one small caustic. Letting the users tweak the number of paths manually is not an option, as it interferes with good usability.

As a part of a general exploration before implementing the caustics solver in Corona, we have collaborated with Saarland university to develop the Lightweight Photon Mapping algorithm [GPGSK18]. This algorithm already addresses some of the issues of VCM mentioned above, and it served as a starting point of our Corona caustics solver implementation. However, a number of specific technical solutions had to be adapted to the challenges and requirements of production rendering as described below.

3.1. Overhead minimization

Let us start with the issue of computational overhead. While the users are willing to accept *some* overhead compared to ordinary path tracing, it needs to be limited to a minimum.

Path sampling techniques. First, as in Lightweight photon mapping, we drop bidirectional connections from full VCM. While using bidirectional connections can somewhat reduce overall noise, it significantly increases overhead. The remaining path sampling techniques can still handle the same light transport phenomena as the bidirectional connections fairly well, so the inclusion of the latter usually does not pay off.

Furthermore, direct connection of photon paths to the camera (i.e. the light tracing technique) can be memory consuming. When using a path tracer with screen-space adaptivity, the direct connection requires a separate image buffer *for each render element* (aka render pass), thus doubling the memory consumption of the entire frame buffer – which can already be fairly high given that architectural rendering often targets 8k or 16k image resolution. For this reason, we drop the direct connections of the photon path as well.

This leaves us with the usual path tracing techniques combined with photon lookups at the camera path vertices. This mix of sampling techniques is still able to efficiently render all light phenomena that can be rendered by vertex connection and merging.

Selective photon mapping. We adopt the idea of selective photon mapping from the Lightweight Photon Mapping algorithm. Consider a full light transport path, connecting a camera to a light source, that can be created either by a photon lookup or by path tracing. If the probability density of generating the path by the path tracing technique is high enough, no photon lookup is necessary [GPGSK18].

To further reduce memory and computation overhead, we only store photons where they may be relevant. First, we only store photons after the first interaction with a specular or highly glossy material, as in the caustics photon map in the original photon mapping technique [Jen96]. Second, we stop the photon path after two consecutive diffuse bounces, because the light distribution then becomes blurred and can usually be well sampled by path tracing.

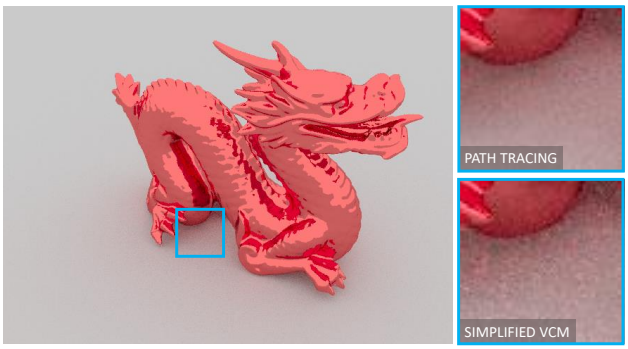


Figure 2: *Dragon statuette made of a highly glossy material illuminated by a uniform environment map. Computing illumination on the floor by path tracing only (top inset) is more efficient than using our simplified VCM, i.e. a combination of path tracing with photon lookups (bottom).*

Environment map emission. Even after considering all the above cases, photons may still be used or stored in some places needlessly. One notable case is uniform environment illumination. Consider a photon path that starts at the environment and bounces off a specular surface before hitting a diffuse floor, as shown in Figure 2. As we can see in the insets, computing illumination on the floor by path tracing produces a less noisy image than by using the combination of photon lookup and path tracing.

To address this issue, we further prune emission of photon paths from the environment. First, we completely suppress photon path emission from a uniform environment, since caustics due to such an environment are easily handled by path tracing. Furthermore, we avoid photon emission from low-value parts of any environment map. Consider the environment map in Figure 3 (left). Emission from such a map would traditionally be based on the intensity of each pixel, resulting in a the emission probability map illustrated in Figure 3, middle. We further reduce the emission probability of the low-intensity parts of the environment map, while increasing the emission from the high-intensity parts (Figure 3, right).

In summary, thanks to the above modifications of the original Vertex Connection and Merging [GKDS12], we have significantly reduced its overhead over path tracing. Our solver is roughly three times faster than the original VCM in scenes featuring many caustics, and the speedup is much higher in scenes featuring small or no caustics. This is achieved by activating the caustics solver only when it is really necessary, while using regular path tracing to handle the rest of the light transport as before.

3.2. Photon guiding

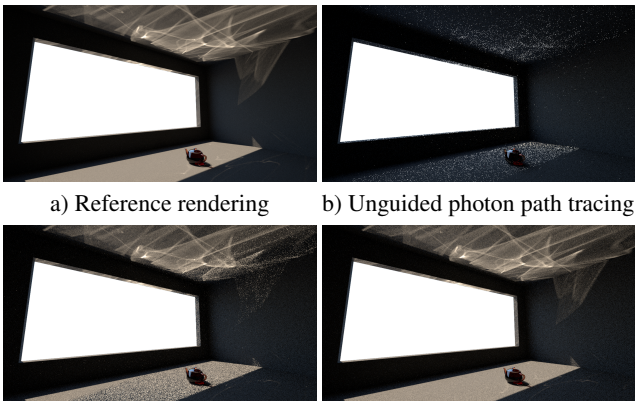
Let us now discuss the issue of photon path guiding in large or complex scenes. Figure 4 shows renderings of an example test scene, where a building with many apartments is lit from the outside. The camera is in one of the apartments and the light coming through the window creates a caustic on the floor. There is a pool next to the building that creates another set of caustics on the apartment's ceiling. Figure 4 (a) shows a reference rendering.

a) Environment map b) Intensity-based emission probability c) 'Sharpened' emission probability map

Figure 3: *The usual approach to determine emission probabilities from an environment map (a) is to use the pixel intensities as the emission probabilities (b). This leads to wasteful sampling of photons that origin from uniform parts of the environment. We lower the emission probability of the low-intensity parts of the environment map, while increasing the emission from the high-intensity parts, ensuring that the generated photons are more likely to generate caustics (c).*

In the same figure (b), we can see that with completely random emission of photon paths, only few will actually reach the visible region, leading to extremely slow convergence. Lightweight photon mapping [GPGSK18] approaches this issue by utilizing adaptive emission. The algorithm emits more photon paths in such directions, where previously emitted photon paths contributed to the image. The emission map for each light source is represented as a grid that is populated based on the results of previous render iterations. The result achieved with this approach is shown in Figure 4 (c). Adaptive emission significantly improves the result, but some parts of the caustics are less converged than other.

Metropolis photon guiding. Based on our previous experience [ŠOHK16], we have decided to utilize Metropolis photon path sampling to guide the photons. To define which paths are important, and should be mutated by the Metropolis sampler, we utilize a data structure which stores the photons visibility [HJ11] to the camera paths. More specifically, the data structure recursively subdivides the scene and marks the visible regions of the scene that contain any camera vertex. When tracing a photon path using Metropolis, we look if any of its vertices lie in the visible regions. If a path has at least one vertex in the visible region, the Metropolis sampler will mutate the path and generate several similar photon paths. On the other hand, if a path has no vertex in the visible region, the path is rejected. This way we ensure that many of the photon paths are generated in the important regions of the scene. We further improve the Metropolis sampling by only marking places in the structure where successful photon lookup occurred in the previous iterations. Using the Metropolis sampling with such definition of important photon paths gives the result shown in Figure 4 (d). We can see that the quality is much better than with any of the previous approaches. Furthermore, we have been able to verify that our Metropolis sampler does not suffer from any flickering artifacts.



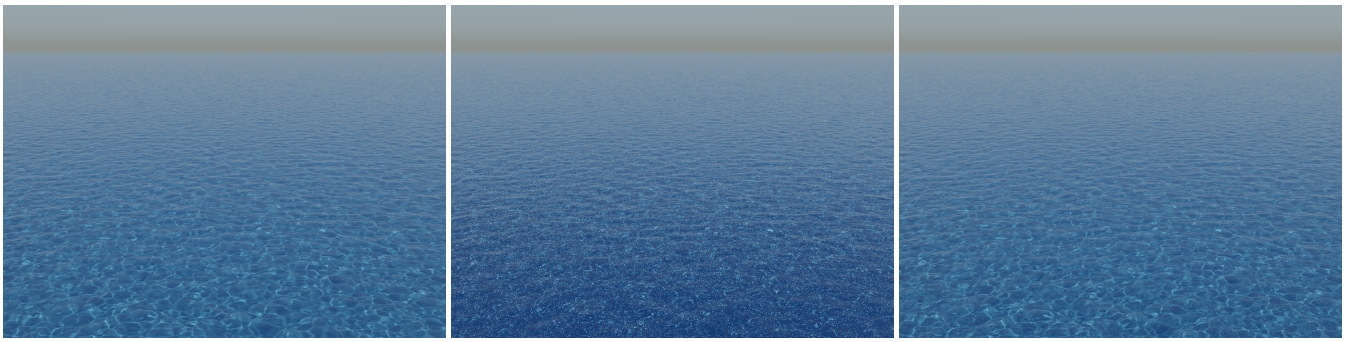
a) Reference rendering b) Unguided photon path tracing

c) Adaptive emission map d) Our Metropolis guiding

Figure 4: *Photon path guiding. a) Reference rendering. b) No photon path guiding. c) Guiding based on an adaptive emission map [GPGSK18]. d) Our Metropolis-based guiding.*

Handling large caustics. There are cases where the distribution of photon generated by the above algorithm is not effective. As an example, consider the ocean scene shown in Figure 5 (left). In this case, the above approach considers the whole visible region as equally important. An equal density of photon paths is therefore distributed across the entire vast visible ocean region, which results in a suboptimal convergence (Figure 5 middle). The regions near to the camera would require denser sampling [GRŠ*17]. We improve the result by decreasing the importance of photon paths according to the distance from the camera. As a result, the Metropolis sampler will generate more photon paths near the camera, and it results to less noisy image, as shown in Figure 5 (right).

Light source selection. The last aspect of our photon path guiding approach is the selection of the light source from which to emit a photon path. Traditionally photon path emission is based on the emitted power (flux) of the light source. Therefore light sources with high power will generate more photon paths, with the overall goal of making all photons carry the same amount of flux [Jen96]. This approach works well for simple scenes featuring a few light sources with similar power, such as in Figure 6 (left). However, adding more light sources including sun and sky, which have much higher power compared to e.g. a light bulb, causes weaker light sources to emit only few photon paths and caustics generated by them to disappear (Figure 6, middle). To solve this issue we add a factor of inverse squared distance to the camera to the light selection probability, as in the case of the definition of important photon paths for the Metropolis sampler. Since the sun and environment are infinitely far away, we can not apply the distance modulation on them. Instead, we initially generate half of the photon paths from the sun and environment and half from the rest of light sources. To robustly handle scenes with a high number of light sources, we further adapt the emission probability of each light source by counting how many photon paths generated from that light source have been able to contribute to the image in previous passes. Our final solu-



a) Reference rendering b) Binary visibility Metropolis sampling c) Our solution

Figure 5: *Handling of large caustics. a) Reference. b) Binary visibility based importance produces suboptimal photon distribution: too many photons are concentrated far away from the camera, while the region near camera is underpopulated. c) Our solution modulates the importance by the inverse squared distance from the camera, producing a more equalized photon distribution.*

tion can robustly handle even the difficult case with thousands of light sources of very different power (Figure 6 right).

3.3. Number of photon paths

The last issue we address is determining the total number of photon paths to trace. Again, we take inspiration from Lightweight photon mapping [GPGSK18]. The algorithm compares the overall image contribution from all the path sampling techniques with the contribution coming only from photon lookup. If the contribution from photon lookup in a given pixel is high enough compared to overall contribution, the lookup is deemed to be useful in that pixel. The number of photon paths is then directly equal to the number of pixels where photon lookup was useful.

This approach, however, does not always work well with Metropolis sampling which requires to send photons in large batches to be effective. For instance, consider a scene that requires good guiding of photon paths and where we want to render just a small region. The number of photons paths is limited by the small region resolution and thus guiding by the Metropolis sampler will be ineffective and the image will converge slowly.

We avoid this issue by setting a fixed maximum number of photon paths, which is not limited by the image resolution. We compute ratio of useful pixels and total number of pixels similarly to Lightweight photon mapping. We then multiply the maximum number of photon paths with this ratio to get the number of traced photon paths. Since such a number can be much higher than the resolution of the image, we further balance the tracing of the paths from the camera and from the light sources across multiple rendering iterations.

More specifically, if we need to trace, say, $4\times$ more photon paths than paths traced from the camera (such as when rendering small render regions), we follow each photon tracing batch by four path tracing batches and utilize the same set of photons for all of them. On the other hand, if the caustics are present only in a small part of the scene, we may trace $4\times$ fewer photon paths than camera paths. In this case, we interleave the photon tracing with path tracing, limiting the overhead of the caustics solver compared to ordinary path tracing.

3.4. Other implementation details

Of course, there is more to our implementation, besides solving the issues of Vertex Connection and Merging. We mention some of the finer details in this section.

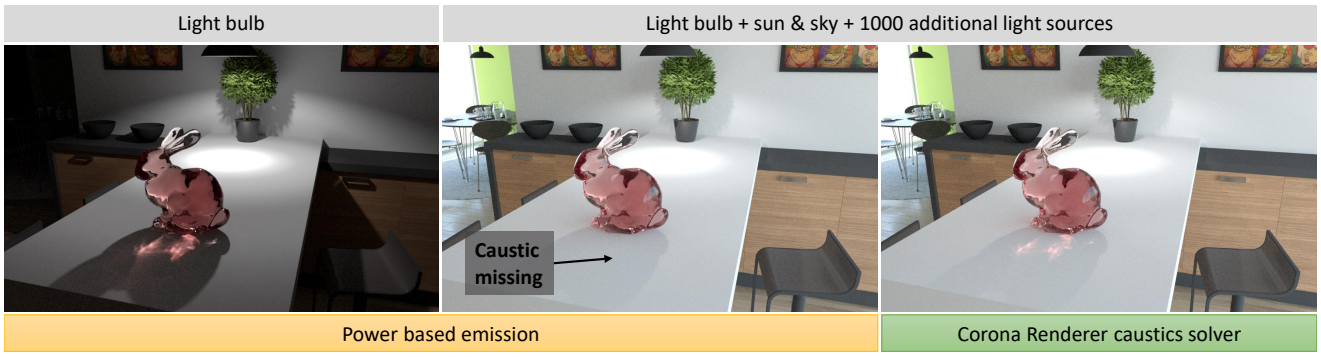


Figure 6: *Light source selection for photon path emission. Power-based emission works well in simple scenes with a few light sources (a) but fails in more complex scenes with numerous light sources including sun and sky (b). Our approach robustly handles such complex cases.*

a) Incorrectly handled motion blur b) Correct motion blur

Figure 7: *Handling of motion blur in our caustics solver. a) Wrong solution, which smears caustics reflected from a car tracked by the camera. b) Correct solution that takes into account time associated with both a camera path and a photon path.*

Motion blur. First such detail is motion blur. Consider a scene shown in Figure 7, where a moving car is being tracked by the camera. When rendering with motion blur, we have to take special care otherwise the caustic created by reflection of the sun from the car would be smeared even though the car is tracked by the camera, see Figure 7 (a). To correctly handle caustics in the presence of motion blur, we independently randomly select an exact time for the camera path and for the photon path. During photon lookup we only consider photons that have similar time as the camera path. This leads to the correct solution, see Figure 7 (b).

Dispersion. Dispersion caustics due to the wavelength-dependent refraction index of a caustics-forming surface are rendered by associating photons with a specific wavelength. Photon lookup then considers only photons with wavelengths similar to the wavelength of the camera path – in case the latter had also undergone wavelength-dependent refraction. The result is shown in Figure 8.

Light mix. Light mix is a popular feature that allows Corona users to interactively adjust light source intensities and colors during rendering. It is critical for good usability that the caustics solver supports this feature. This is trivially achieved by keeping track of the light source from which each photon originates.

Caustics control. Finally, to allow users to have better control over the caustics, we give them an option to render the caustics in a separate image buffer for better control in compositing.

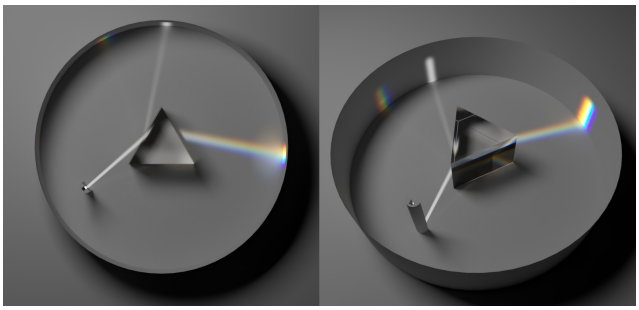


Figure 8: *Dispersion caustics due to the wavelength-dependent refraction index of a caustics-forming surface. Image courtesy of Rakete GmbH Munich.*

	Pool scene	Ring scene	Glasses scene
Image resolution	1440×810	1920×1080	1280×720
#iterations	80	100	100
Time - path tracing	21 min	4.8 min	3.5 min
Time - caustics solver	38 min	9.4 min	7 min
#photon paths/iter	3.8M	1.6M	2.2M

Table 1: *Settings and statistics for scenes in Figures 1,9.*

4. Results

We have tested our new caustics solver for Corona Renderer on a PC machine with an AMD Ryzen Threadripper 1950X at 3.40 GHz with 32 GB RAM using 32 logical cores. We compare the performance of our new caustics solver with path tracing with next event estimation (the default rendering algorithm in Corona Renderer). All our comparisons are equal-iteration comparisons, see Table 1 for rendering statistics.

Figure 1 shows the **pool** scene, where path tracing fails to resolve the caustics on the bottom of the pool and also caustics reflected on the left wall. The result of path tracing has further clamped high contributing samples to avoid fireflies and allows unhindered transmission through water surface to sufficiently lit the pool interior. On the other hand, the caustics solver delivers correct solution without any fakes with overhead compared around 81% to path tracing.

The **ring** scene (Figure 9, top) shows both direct and indirect caustics created by the reflection from the ring. Again, unlike our new caustics solver, path tracing fails to resolve the caustics which must be clamped to avoid fireflies. In this scene the overhead is approximately 95%.

The **glasses** scene (Figure 9, bottom) illustrates that path tracing that uses unhindered transmission through glass can generate fake caustics, but they are very different from the correct ones produced by our caustics solver. In this scene the overhead is roughly 99%.

Overall we can see that path tracing fails to resolve the caustics without resorting to fake solutions in all the presented scenes, while our caustics solver delivers the correct solution with less than 100% overhead.

5. Conclusion

We have presented a new caustics solver implemented in Corona Renderer version 4. While based on the rather heavy-weight Vertex Connection and Merging algorithm, the judicious design of the solver means a fairly low overhead over path tracing. Our solution is fully automatic, meaning the user does not have to setup anything by hand. The solver will automatically guide the required photon paths toward the important visible regions and determine the required number of photon paths. Finally, the solver is compatible with all other Corona settings and behaves the same way as ordinary path tracing in the parts of the scene without caustics.

While shipped as a part of the product release, we consider the caustics solver implementation as an initial attempt. Future work includes fully automatic handling of caustics in participating media, where a carefully selected subset of the unified photons, beams

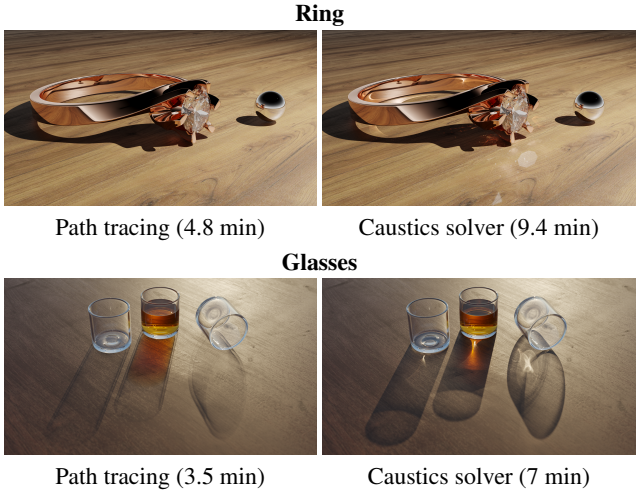


Figure 9: *Comparison of path tracing with fake handling of caustics and our caustics solver in Corona Renderer. The ring model (top) is courtesy of Free_opinion kindly provided by Turbosquid.*

and paths (UPBP) algorithm can serve as a good basis [KGH*14]. The Metropolis photon sampling still exhibits some undesirable correlation artifacts in the image and more work is needed to eliminate those.

Acknowledgments

Many thanks to Corona Renderer users for stress-testing the caustic solver throughout its development and for providing some of the test scenes. This work was partially supported by the Czech Science Foundation grant 19-07626S.

References

[BAC*18] Burley B., Adler D., Chiang M. J.-Y., Driskill H., Habel R., Kelly P., Kutz P., Li Y. K., Teece D.: The design and evolution of Disney's Hyperion renderer. *ACM Trans. Graph. 37*, 3 (2018), 33:1–33:22. 1

[CFS*18] Christensen P., Fong J., Shade J., Wooten W., Schubert B., Kensler A., Friedman S., Kilpatrick C., Ramshaw C., Bannister M., Rayner B., Brouillat J., Liani M.: Renderman: An advanced path-tracing architecture for movie rendering. *ACM Trans. Graph. 37*, 3 (2018), 30:1–30:21. 1

[FHL*18] Fascione L., Hanika J., Leone M., Droske M., Schwarzhaupt J., Davidovič T., Weidlich A., Meng J.: Manuka: A batch-shading architecture for spectral path tracing in movie production. *ACM Trans. Graph. 37*, 3 (2018), 31:1–31:18. 1

[GIF*18] Georgiev I., Ize T., Farnsworth M., Montoya-Vozmediano R., King A., Lommel B. V., Jimenez A., Anson O., Ogaki S., Johnston E., Herubel A., Russell D., Servant F., Fajardo M.: Arnold: A brute-force production path tracer. *ACM Trans. Graph. 37*, 3 (2018), 32:1–32:12. 1

[GKDS12] Georgiev I., Křivánek J., Davidovič T., Slusallek P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph. 31*, 6 (2012), 192:1–192:10. 1, 2, 3

[GPGSK18] Grittmann P., Pérard-Gayot A., Slusallek P., Křivánek J.: Efficient caustic rendering with lightweight photon mapping. *Computer Graphics Forum 37*, 4 (2018). EGSR '18. 1, 3, 4, 5

[GRŠ*17] Gruson A., Ribardière M., Šik M., Vorba J., Cozot R., Bouatouch K., Křivánek J.: A spatial target function for Metropolis photon tracing. *ACM Trans. Graph. (TOG) 36*, 1 (2017), 4. 2, 4

[HDF15] Hanika J., Droske M., Fascione L.: Manifold next event estimation. *Comput. Graph. Forum 34*, 4 (2015), 87–97. 2

[HEV*] Herholz S., Elek O., Vorba J., Lensch H., Křivánek J.: Product importance sampling for light transport path guiding. *Computer Graphics Forum 35*, 4, 67–77. 2

[HJ09] Hachisuka T., Jensen H. W.: Stochastic progressive photon mapping. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 141. 1, 2

[HJ11] Hachisuka T., Jensen H. W.: Robust adaptive photon tracing using photon path visibility. *ACM Trans. Graph. (TOG) 30*, 5 (2011), 114. 2, 4

[HOJ08] Hachisuka T., Ogaki S., Jensen H. W.: Progressive photon mapping. *ACM Trans. Graph. 27*, 5 (2008), 130:1–130:8. 1, 2

[HPJ12] Hachisuka T., Pantaleoni J., Jensen H. W.: A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 191. 1, 2

[Jen95] Jensen H. W.: Importance driven path tracing using the photon map. In *Rendering Techniques 95*. Springer, 1995, pp. 326–335. 2

[Jen96] Jensen H. W.: Global illumination using photon maps. In *Rendering Techniques 96*. Springer, 1996, pp. 21–30. 1, 2, 3, 4

[KCSG18] Kulla C., Conty A., Stein C., Gritz L.: Sony Pictures Imageworks Arnold. *ACM Trans. Graph. 37*, 3 (2018), 29:1–29:18. 1

[KGH*14] Křivánek J., Georgiev I., Hachisuka T., Vévoda P., Šik M., Nowrouzezahrai D., Jarosz W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 103. 1, 7

[KKK*18] Křivánek J., Karlík O., Koylazov V., Jensen H. W., Ludwig T., Chevallier C.: Realistic rendering in architecture and product visualization. In *ACM SIGGRAPH 2018 Courses* (2018). 1

[KKN*18] Keller A., Křivánek J., Novák J., Kaplanyan A., Salvi M.: Machine learning and rendering. In *ACM SIGGRAPH 2018 Courses* (2018), ACM. 2

[MGN17] Müller T., Gross M., Novák J.: Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum (EGSR '2017) 36*, 4 (2017), 91–100. 2

[MMR*18] Müller T., McWilliams B., Rousselle F., Gross M., Novák J.: Neural importance sampling. *arXiv preprint arXiv:1808.03856* (2018). 2

[PLW98] P. Lafortune E., Willems Y.: Bi-directional path tracing. In *Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics)* (1998). 1, 2

[ŠK19] Šik M., Křivánek J.: Survey of Markov chain Monte Carlo methods in light transport simulation. *IEEE Transactions on Visualization and Computer Graphics* (2019). 2

[ŠOHK16] Šik M., Otsu H., Hachisuka T., Křivánek J.: Robust light transport simulation via metropolised bidirectional estimators. *ACM Trans. Graph. 35*, 6 (2016), 245:1–245:12. 2, 4

[VG95] Veach E., Guibas L.: Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. 1995. 1, 2

[VG97] Veach E., Guibas L. J.: Metropolis light transport. In *ACM SIGGRAPH '97* (1997), pp. 65–76. 2

[VKŠ*14] Vorba J., Karlík O., Šik M., Ritschel T., Křivánek J.: On-line learning of parametric mixture models for light transport simulation. *ACM Trans. Graph. (TOG) 33*, 4 (2014), 101. 1, 2