Vision, Modeling, and Visualization (2022)
J. Bender, M. Botsch, and D. Keim (Eds.)

Real-Time Caustics Using Cascaded Image-Space Photon Tracing

Krittin Meenrattanakorn and Martin Lambers

University of Siegen, Germany

Figure 1: *Ray-traced caustics in the flooded Sponza scene using only image-space computations. We achieve refractive caustics beneath the water, as well as reflective caustics on the wall and the ceiling without depending on hardware-accelerated ray tracing in recent GPUs.*

Abstract

Caustics are formed when transparent or specular materials focus many light rays onto confined areas. Such effects are among the most challenging optical effects to render. Typical path tracing algorithms need many light ray samples to form realistic caustic highlights and/or clever strategies to sample the most relevant light paths. This paper presents an approach for rendering caustics from one specular bounce at an interactive performance by using cascaded image-space photon tracing based on both G-buffer (camera perspective) and Reflective Shadow Maps (light source perspective). A denoiser is applied to remove Monte Carlo noise before applying the caustics texture. Our results demonstrate caustics with more realistic and precise details compared to state-of-the-art Caustics Mapping, at real-time frame rates even on moderately powerful GPUs.

CCS Concepts

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

1. Introduction

Caustic is an optical phenomenon in which light is focused onto a surface in a non-uniform way. It is caused by either reflection or refraction at a non-planar surface leading to unequal distribution of photons onto the surface. A typical example of refractive caustics is the light patterns that appear on the bottom of a swimming pool as a result of sunlight hitting the waves on the water surface. Surfaces above the water surface may additionally exhibit reflective caustics, for example on the pool walls or a swimmer's face.

Rendering caustics is typically done via ray tracing approaches that use biased Monte Carlo method [Jen96] [GKDS12] [KD13] [ZGJ20]. The main challenge is that sampling light paths relevant to caustics often has a low probability, especially when the light path contains a specular-diffuse-specular (*SDS*) subpath. For instance, a camera ray hits the water surface, then the diffuse bottom of the pool, then the water surface again before the light source is reached. However, such techniques are not usable in real-time rendering, where rasterization is still the dominant method of image generation. Direct illumination evaluates only the first bounce of the camera ray, whereas caustics rendering requires at least two bounces. Approximation techniques have limitations in that they are either phenomenological and lack a physical basis, or fail to reach the quality levels of ray tracing based techniques. Modern powerful GPUs with hardware-accelerated ray tracing can alleviate this to a certain degree, but at high hardware costs.

DOI: 10.2312/vmv.20221212

In this paper, we present a caustics rendering approach that works entirely in image space and does not require GPU-accelerated ray tracing support. We trace light using the views from the light source and the camera, and with the help of a denoiser, full paths contributing to the caustics are constructed, even in the challenging *SDS* case using existing real-time reflection and refraction techniques. We demonstrate our Cascaded Image-Space Photon Tracing approach using an example scene in which a water surface causes both refractive and reflective caustics. Our approach is more realistic and precise than existing real-time caustics approximations, and runs at interactive frame rates even on moderately powerful GPUs.

2. Related Work

Rendering caustics using a pure ray tracing pipeline is straightforward, and specialized methods mainly optimize sampling strategies. Instead, we focus on rendering techniques that rely on the rasterization pipeline, as well as the hybrid approaches that utilize ray tracing in some parts of their algorithm.

2.1. Fake Caustics

Caustics are still a challenging problem in real-time rendering. There are many attempts to reproduce the caustics by either faking or approximating them. Using a texture as a caustics map [Sta96] is one of the techniques that can be utilized without much effort. The caustics texture is generated by evaluating a periodic wave equation using Fast Fourier Transform (FFT), constructing a quad, and projecting it onto the texture. By sampling this caustics map and applying it to the surface beneath the water, the caustics are rendered very fast without any intensive computation. Animating the caustics can also be achieved by preparing a set of animated caustics maps and switching the texture over time. However, the rendered caustics themselves do not reflect the real light transport given the rendered water surface.

2.2. Image-Space Caustics

For directional lights, spotlights, and point lights, the first hit of a photon can be evaluated by Reflective Shadow Maps (RSMs) [DS05], which typically store depth, normal, and flux. Thus, several techniques exploit light view information in RSMs to evaluate the second photon hit by either a static displacement [Sou05] or an iterative process [SKP07]. Caustics Mapping [WD06] [SKP07] [WN09] splats photon points at its second hit onto a light-space texture called Caustics Map, and samples this map by reprojection as Shadow Mapping does to render the caustics in a camera view. A triangle formed by a reflective or refractive medium's vertices can also be warped onto a receiver surface in accordance with the ray direction [EAMJ05]. With this method, the point splatting is not required anymore, and more coherent-shape caustics are rendered as a result. Another method is to construct a height field to form a water surface, instantly gather the photons from a refraction model, and convolve the final caustics map [YK09]. Note that the receiver has to be a flat plane, and the water cannot be very turbulent. All of these methods achieve a real-time frame rate; however, they have a common limitation: the caustics are rendered if and only if they are visible to the light source. The caustics outside the light view frustum, as well as those being occluded in the light view, are not addressed. Splatting photons in the camera space instead [DS06] [NW09] [NW10] makes the caustics independent from the light view, since the light is scattered to the area where it is actually visible to the camera. However, the visibility between the first and the second hits is still ignored, leading to a light leak.

2.3. Ray-Traced Caustics

CPU-GPU cooperation [ML09] is a way to offload an intensive thread-divergent task like ray tracing from the GPU. The rays are generated using the first hit information from the RSMs, and are sent to a multi-core CPU to trace the rays. After finding a hit point or a miss, the CPU sends the tracing results back to the GPU for further shading. This approach can achieve a real-time frame rate; however, it burdens the CPU, which can be assigned other intensive tasks in the meantime such as pathfinding, and also other processes in an operating system. After the introduction of the hardware-accelerated ray tracing feature in recent GPUs, many pure ray-tracing approaches are tweaked to run on these GPUs instead. This also applies to Photon Mapping. Smal et al. [SAB19] demonstrates that they can explicitly trace photons in world-space, and then calculate a radiance to a screen by photon splatting. In contrast, Kim et al. [Kim19] denoise the photons instead of splatting. Without a doubt, the final result is the most promising and physically correct among other real-time approaches; however, both depend on a GPU with hardware-accelerated ray tracing support to maintain real-time performance.

Rather than tracing expensively in world space, Sousa et al. [SKS11] proposed an alternative approach to trace the ray in image (or screen) space, while being able to query the visibility via a depth buffer. McGuire et al. [MM14] later enhanced this technique by changing the way the ray marches, such that it does not suffer from over-sampling and under-sampling problems. This method is applied mostly to the screen-space reflection and refraction effects [MM14] [Sta15], where the rays tend to march in the same direction, leading to texture cache efficiency. Yet there is no such technique for caustics that operates in screen space.

3. Cascaded Image-Space Photon Tracing

Our algorithm aims at rendering the caustics induced by a single specular surface by assuming that the receiver surface is diffuse, where supported light paths are shown in Figure 3. A high-level overview is shown in Figure 4, where our algorithm is integrated into a deferred shading pipeline. Prerequisite data consists of RSMs [DS05] of the light view, and G-Buffer [ST90] of the camera view. These buffers contain the necessary information to identify the first ray hit from the source: a world-space position, a surface normal, and surface material attributes. Note that buffer generation is separated into two passes: opaque and transparent (refractive). In case of refractive caustics, a ray needs a view of the geometries behind the refractive object, so a separated depth buffer exhibiting only opaque objects is required. Next, photons are traced from the first hit and save their second hit in a screen-size texture, being ready to be composited with the main render result. After that,

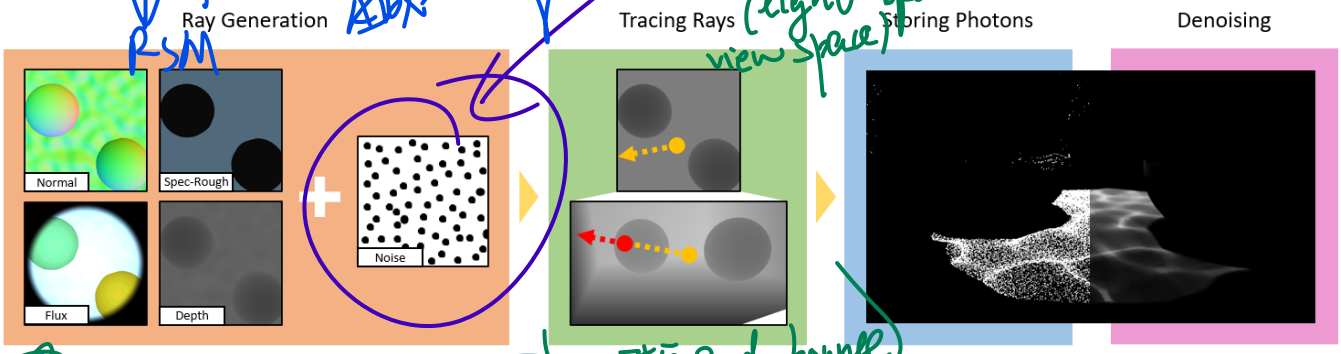

Figure 2: *An overview of Cascaded Image-Space Photon Tracing. Pseudo-random numbers are used to determine sampling points to sample information from RSMs, and construct a ray. The ray is then traced through two depth buffers from the light source view to the camera view. After that, the photons are stored in a screen-sized texture (photon map), and the denoiser operates on this texture to produce the final result.*

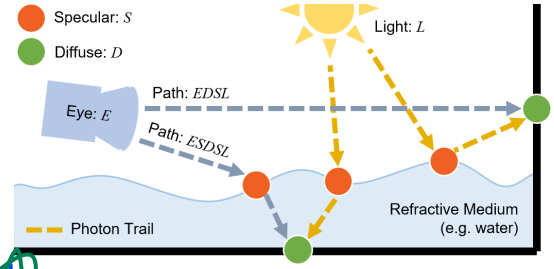

Figure 3: *Supported light paths are of directly-visible caustics (EDSL) and those being visible through specular surface (ESDSL).*

post-processing effects can be used to enhance a visual appearance such as screen-space reflection and refraction [MM14] [Sta15], depth of field, bloom, and lens distortion. Temporal anti-aliasing (TAA) [Kar14] is recommended since our algorithm depends on a temporal filter (denoiser).

A photon tracing procedure is given in Figure 2. Starting at the RSMs, depth, normal, flux, and material attributes are sampled. For Physically-Based Rendering (PBR) workflow [Hut21], the only relevant attribute is roughness, which describes how incoming lights are scattered. It is usually packed with a specular color to form a four-channel pixel. Lastly, to trace the ray, the depth buffer from the opaque-pass RSMs is needed, as well as another from the opaque-pass G-Buffer. The reason for using the camera-view depth buffer will be explained later in the tracing rays subsection.

3.1. Ray Generation

All RSMs texels are candidates for ray origins for photon tracing. To reduce the computational costs, sampling points are chosen randomly using blue noise, a low-discrepancy sequence of pseudo-random numbers in $[0,1]^2$, as RSMs texture coordinates. Note that the sampled flux must be scaled by the ratio of the RSMs resolution to the total number of samples. This is to preserve the radiance intensity of the final caustics regardless of the number of samples.

The set of randomly chosen sampling points contains irrelevant

Figure 4: *The algorithm applied to a deferred shading pipeline. Green denotes geometry passes, and orange denotes effect passes based on G-Buffer and RSMs data.*

ones that must be removed. Since photons that influence caustics are considered exclusively, the roughness of the material is taken into account to identify whether the surface is specular or not. Empirically, the roughness cutoff is set to 0.3 out of 1.0. If it exceeds this cutoff, it means the surface is diffuse, which is not likely to enforce bouncing of the light ray to a specific direction as a specular surface would. Another criterion is the current luminance calculated using the flux. If it is too low, it is useless to trace that photon since it is likely to contribute nothing to the final caustics. The luminance cutoff parameter is set to 0.04 in our experiments.

After a valid sampling point is retrieved, the ray is set up by sampling relevant information from the RSMs using the corresponding texture coordinate. The ray is defined as followed:

$$r(t) = o + \vec{d}t \tag{1}$$

For the origin o, the view-space position is reconstructed using the depth value and the projection matrix [Kli08]. For the bouncing direction $\vec{d}$, whether the light ray is reflected or refracted depends on the *Fresnel factor* F, which takes IOR of both source and destination mediums into consideration. This factor is interpreted as a probability used for discrete inverse transform sampling: ranging between 0 and 1, if the noise value exceeds this factor, the light ray is refracted. After o and $\vec{d}$ are found, the ray traverses by parameterizing t in the next section.

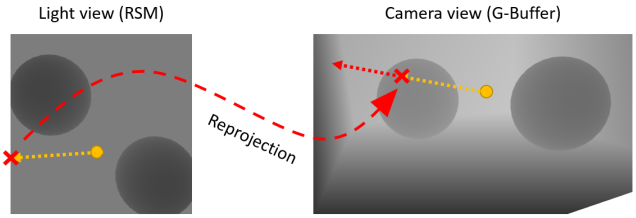

Figure 5: *Reprojection upon the termination of the ray in the light view. The ray is traced further in the camera view to find an exact hit.*

3.2. Tracing Rays

The key advantage of our algorithm in photon tracing is that *cascaded* image spaces are exploited: from the light view to the camera view. As illustrated in Figure 5, no matter if a hit or a miss is found in the light view, the last ray end position is then *reprojected* to the camera view to find an exact hit. If it is a miss, either in the light view or after the reprojection, the ray has a second chance to march forward until the exact hit is found in the camera view. This increases a chance of finding the hit compared to using a single view as other frustum-bound techniques [EAMJ05] [WD06] [SKP07] [WN09] [YK09]. The ray is traced using the standard image-space ray marching technique [MM14] [Sta15] that traverses through pixels, evaluates t value at each step, and checks the ray visibility using the depth map. After finding the exact hit, it is verified by sampling the normal of the hit surface. The incoming direction $\vec{d}$ must be consistent to the surface normal $\vec{n}$: $\langle -\vec{d}, \vec{n} \rangle \geq 0$.

To improve the reprojection quality, the most precise final t value is needed such that, when substituting in the ray equation, it is exactly the position where the ray terminates in the light view space. A linear gradient in depth is assumed between two adjacent pixels, and a linear equation is constructed using the clip-space depth value z_r where the ray steps, and the sample depth value z_s of the geometry beneath. If $z_r > z_s$, the tracing is terminated, and the last position is where the ray penetrates the geometry beneath. As illustrated in Figure 6, given the last pixel at $t = t_n$ where the ray end is still visible, the view-space ray end depth $V_{z,r0}$ and the view-space sample depth $V_{z,s0}$ (blue cross) are evaluated. Analogously, on the next pixel at t_{n+1} where the ray starts to penetrate, another view-space ray end depth $V_{z,r1}$ and view-space sample depth $V_{z,s1}$ (pink cross) are evaluated. The view-space sample depth is calculated from the depth value z_s using the same method as reconstructing the ray origin o [Kli08]. Using the view-space depth instead of the clip-space depth prevents a depth precision problem of objects located far from the camera, which leads to an ambiguity between the two depth values. Given a variable $u \in [0,1]$ that linearly maps $t \in [t_n, t_{n+1}]$, two line segments are defined as followed:

$$\begin{cases} V_{z,r} = (V_{z,r1} - V_{z,r0})u + V_{z,r0} \\ V_{z,s} = (V_{z,s1} - V_{z,s0})u + V_{z,s0} \end{cases} \tag{2}$$

given line segments for the ray $V_{z,r}$ and the geometry beneath $V_{z,s}$. After that, the intersection point of these two lines is evaluated,

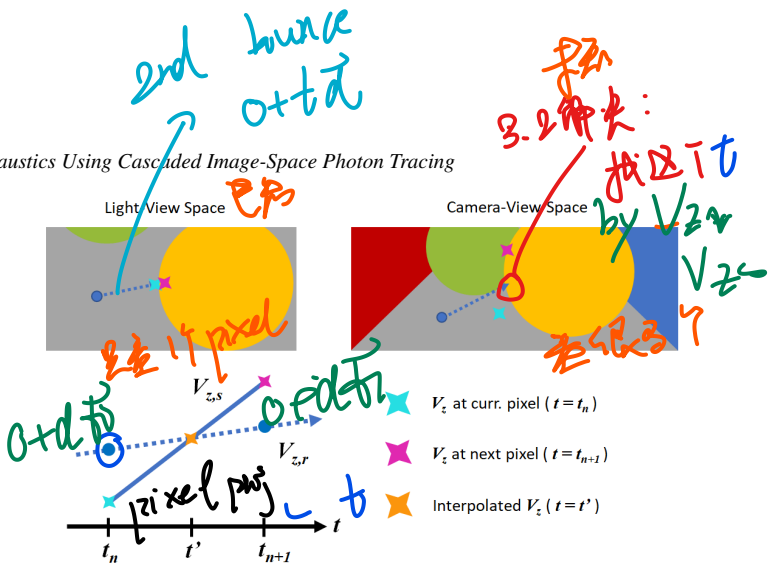

Figure 6: *Interpolation to find an exact termination point before reprojection. Using a pair of view-space depth from the ray $V_{z,r}$ and the scene geometry $V_{z,s}$, we construct a linear equation system and find the intersection point.*

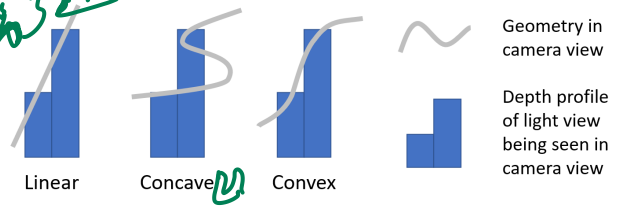

Figure 7: *After reprojection, there are three possible cases that influence whether the photon can be traced further in the camera view or not.*

which gives u' that is mapped back to $t' \in [t_n, t_{n+1}]$. This interpolated t' is the value fed to the ray equation $r(t)$ to get the position being reprojected.

Note that there is a concern over the reprojection against the scene geometry. There are three cases to consider as shown in Figure 7. According to Figure 6, if those two points, the blue cross and the pink cross, have a linear transition (i.e., a plane) in the camera view, the reprojected ray end will lie on a geometry surface, and it is likely to be an exact hit instantly. If those points have a concave transition as shown in Figure 6, the reprojected ray end will float in the air. In this case, the ray can still be traced further in the camera view. But if those points have a convex transition, the reprojected ray end will be occluded and discarded at the start. This last case may result in a loss of quality if the RSMs resolution is too low to preserve exact surface gradients.

3.3. Irradiance Estimation

After the exact hit point is identified, the irradiance caused by the traced photon is calculated. According to Kim et al. [Kim19], the world-space area A_p of a pixel p is

$$A_p = \left(\frac{2d \tan\left(\frac{\theta_x}{2}\right)}{w} \right) \left(\frac{2d \tan\left(\frac{\theta_y}{2}\right)}{h} \right)$$

given the screen's width w and height h, view frustum angles θ_x and θ_y in x- and y-axis respectively, and the distance d from the eye (camera) to the point in world space that the pixel covers. Based on this information, the irradiance is calculated using the flux value Φ carried by the photon:

$$I_p = \frac{\Phi}{A_p}\langle -\vec{d}, \vec{n} \rangle \tag{3}$$

Since more than one photon can terminate at the same screen pixel, to avoid race conditions, all photons are saved into a buffer as a tuple of clip-space coordinates and the calculated irradiance. This buffer is then used as a vertex buffer to render each photon as a point primitive with size 1, resulting in a 2D photon map as shown in Figure 2. Alternatively, writing directly to the texture without point rendering is possible on GPUs with atomic texture write support.

Lastly, the sparse photons are gathered to form the caustics by denoising (i.e., blurring) the photon map. The objective of the denoiser is to estimate the density of the photons in a particular pixel, which accommodates connecting subpaths traced from the light source to the camera. Since the number of samples is limited and yields the noisy photon map, the denoiser removes noise by *gathering* [MLM13] nearby photons and accumulating their contribution at each pixel. Sampling the denoised photon map pixel with known contribution is comparable to finding a full caustics light path. In this work, the Spatiotemporal Variance-Guided Filter (SVGF) [SKW*17] is used, which filters the image spatially and temporally. This denoiser is ideal for a sequence of images produced by Monte Carlo Path Tracing [Kaj86]. It requires the G-Buffer's normal, depth, and a motion vector to operate its own history buffers. After the denoising, the final caustics texture is ready to be composed with the main render result. Assuming that the receiver surface is diffuse, a radiance towards the camera is evaluated by considering *Lambertian reflectance*.

4. Results and Evaluation

Two scenes are used for evaluation: Cornell Box and Sponza. Both of them are flooded to demonstrate refractive and reflective caustics. To emphasize caustics, the IOR of water is set to 2.4 (diamond) in the Cornell Box scene. In the Sponza scene, it remains 1.33. The screen resolution is 1920x1080, and the RSMs resolution is 2048x2048 to support up to 2 million photons.

The original Caustics Mapping [SKP07] is the main approach to be compared against ours. Since it depends on water's vertices to initialize the photons, this approach is not evaluated in the Sponza scene since the water surface is implemented by a simple plane with normal mapping, so there is no vertex to be evaluated as a photon starting point. To investigate correctness, Path Tracing is used as a reference, which is rendered using LuxCore renderer plugin for Blender [Lux21] that, in contrast to the built-in Cycles renderer, renders caustics more efficiently including the *SDS* subpath. 16 samples per pixel (both from a camera and a light source) are used, and the result image is denoised by Open Image Denoiser (OIDN) [Int21].

4.1. Quality

The most significant advantage of our approach over the others is that it can exhibit caustics outside of the light frustum. As shown in the Cornell Box scene in the first row of Figure 8, the reflective caustics on the wall are rendered only when using our approach and Path Tracing. Nevertheless, the caustics shape is less precise compared to the refractive caustics. Regarding the refractive caustics on the floor, our approach and Caustics Mapping achieve identical to Path Tracing. However, the edges are significantly sharper in our approach, and the intensity gradient is smoother than with Caustics Mapping. These are advantages of denoising against splatting for photon density estimation. One drawback is that the contrast at the outermost boundary is stronger compared to other methods. This is because of too much density of photon points at the border, which leads to an outstanding bright region instead of fading from inside to outside. Meanwhile, Caustics Mapping is not affected because it utilizes photon splatting instead of denoising, which distributes irradiance better to adjacent pixels in a splatting area. Further results in the Sponza scene are shown in the second row of the same figure, where the caustics containing *SDS* light subpath are more explicitly exhibited through the water surface via the standard screen-space refraction. Compared to Path Tracing, the result is qualitatively plausible.

Comparison using different numbers of photons is shown in Figure 9. Three numbers of photons are selected for evaluation: half million, one million, and two million. The results of using half-million photons in both the Cornell Box and the Sponza scenes are not much different from the results of using two million photons. However, in the area where the photons are quite sparse such as the wall beside the lion in the Sponza scene, the caustics are blurrier in the case of half-million photons.

4.2. Performance

The Sponza scenes with different numbers of photons configured are evaluated and compared. The GPU used for testing is AMD Radeon RX6500XT at 2.815 GHz boost clock with 4GB GDDR6 video memory. Note that it is configured as an external GPU: a host system and the GPU is connected via Thunderbolt 3 cable at PCI express 3.0 x4 throughput. Thus, it is expected that the achieved performance is worse than attaching directly to the host system with its most-capable PCI express 4.0 x4 interface.

The performance is shown in Table 1, where our approach is decomposed into different parts. At half-million photons where the quality is not significantly different from using two million photons, our approach takes 22 ms, which is within the budget of the typical smooth frame rate at 30 fps, or 33.333 ms. It is clear that the largest contribution to the execution time is tracing rays in every configuration, and it grows proportionally to the number of photons used. However, storing photons and denoising are not significantly affected by the increasing number of photons. A more apparent impact on storing photons would likely be an increased memory footprint.

Note that the implementation has not been optimized yet. Several aspects can contribute to the faster execution of our approach. First, Image-Space Ray Tracing traverses pixel-by-pixel in the depth

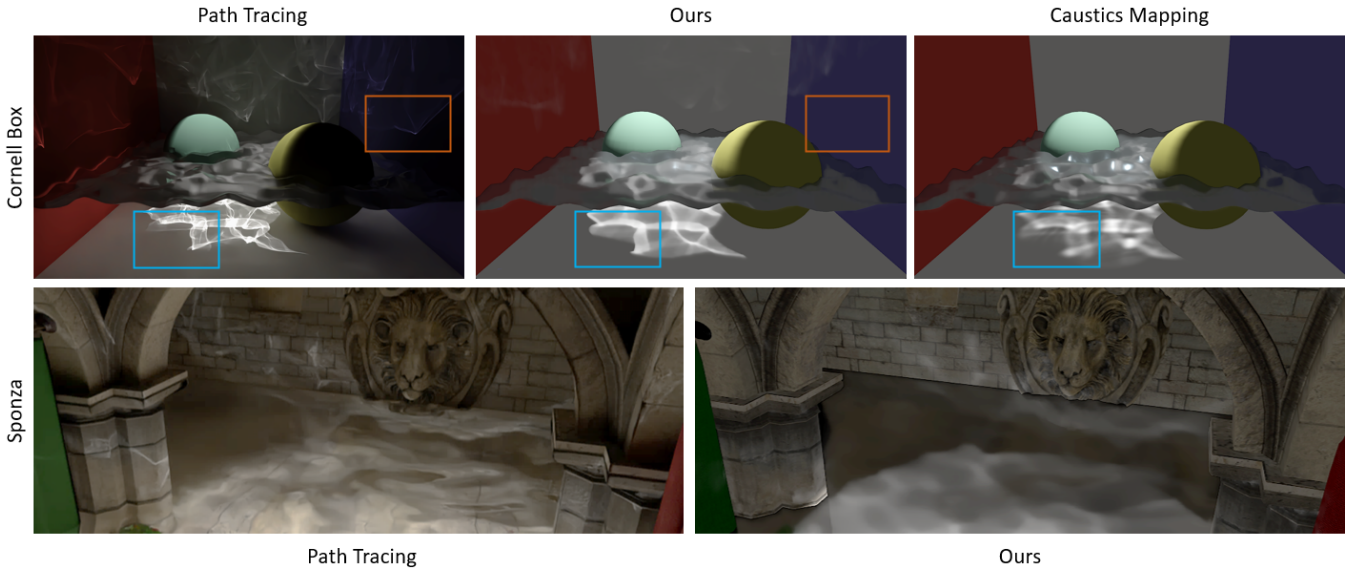

Figure 8: *Quality comparison to different approaches. Path Tracing result in the Cornell Box scene (top-left) is depicted as a reference. Comparing our approach (top-center) to Caustics Mapping (top-right), its achievable caustics are more precise. Moreover, reflective caustics are visible outside of the light frustum, in contrast to Caustics Mapping. Path Tracing result in the Sponza scene (bottom-left) is further demonstrated, where the caustics below the water (via screen-space refraction) are more explicitly visible, as well as the caustics on the wall and the pillar compared to the result in the Cornell Box scene. Note that the contrast at the outermost boundary is stronger than the others (blue rectangles), and the caustics in some regions are not exposed in our result compared to Path Tracing (orange rectangles) due to the photon trail being occluded in the camera view.*

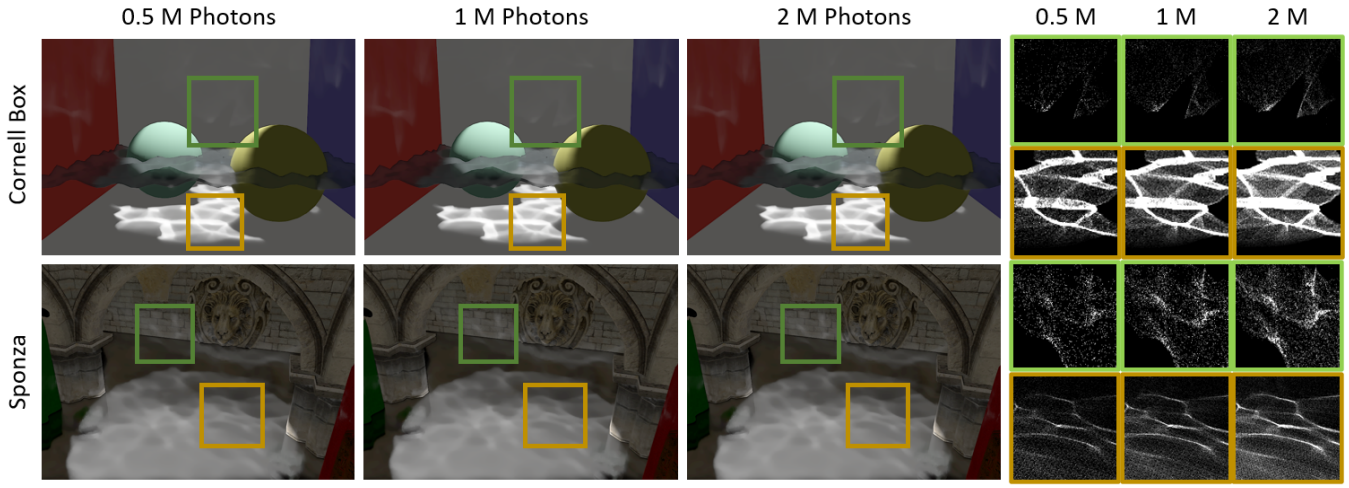

Figure 9: *Quality comparison using different numbers of photons in the Cornell Box and the Sponza scenes. Two areas are emphasized in each scene as shown in a colored rectangle (green and yellow). The more photons are sampled, the more quality our approach gives, as the density of photon points is shown on the rightmost three columns. Nevertheless, the brightness of the caustics is preserved in every configuration, since the flux is scaled in accordance with the number of photon samples.*

Table 1: *Performance comparison using different numbers of photons. The algorithm is decomposed into three parts: tracing rays, storing photons, and denoising. In each part, execution time in ms. and its percent of contribution are shown.*

# Photons	0.5M		1M		2M	
	t (ms.)	ctb.	t (ms.)	ctb.	t (ms.)	ctb.
Tracing	11.67	53%	23.52	68%	43.20	77%
Storing	0.39	2%	0.61	2%	2.02	4%
Denoising	9.94	45%	10.55	30%	10.88	19%
Total	22.00	-	34.68	-	56.10	-

buffer, which takes the number of iterations up to the number of pixels along the image's diagonal. It can potentially be accelerated by Hierarchical Image-Space Ray Tracing [MM14] [Sta15], which traverses a ray in the mipmaps of the depth buffer and skips many intermediate pixels on the base-level texture. Second, in a particular compute unit (i.e., a warp in CUDA, or a subgroup in Vulkan), a sampling coordinate for one thread is gathered randomly. If the threads in the same compute unit have their sampling coordinate too far from each other, there will be massive texture cache misses since their rays traverse in diverged directions, resulting in a drastic performance penalty. This can be avoided by sorting the sampling coordinates such that the threads in the same compute unit start tracing closer to each other.

4.3. Limitations

The main noticeable drawback of our approach is the loss of photons in certain views. If a light path is not contiguously visible in the light and camera views, the caustics are lost. This inevitable issue happens for all screen-space techniques, which take only the scene information being visible to the frustum into consideration. As shown in Figure 8, the caustics on the blue wall existing in Path Tracing do not appear in our case. The related photons cannot be traced after the reprojection, since the yellow ball occludes the photon trail on the camera view.

Panning the camera, there is still low-frequency noise of the reflective caustics that form from a very sparse photons area (e.g. a wall in the Cornell Box scene as shown in Figure 10). Meanwhile, the refractive caustics that form from a much more dense photons area do not suffer from this temporal instability. However, this noise does not exist if the camera is still, even though the caustics are animated. Improving the denoiser, especially where temporal accumulation is performed, would solve this problem.

Another limitation is that if the receiver surface is not diffuse, the caustics being visible through a refractive surface is no longer physically plausible. Since the photon map does not store an incident vector, a radiance cannot be evaluated correctly due to specular reflectance.

Finally, our approach supports a limited set of light types. Since it depends on RSMs, only directional lights, spotlights, and point lights are applicable. Area lights are not possible since there are infinite points of light origin on its surface. Thus, defining a view frustum from the area light is impossible.

Figure 10: *Temporal instability upon camera panning exposed as low-frequency noise in Cornell Box (left) and Sponza (right) scenes.*

5. Future Work

The quality of the caustics can be further improved in several ways. First, sampling efficiency can be improved by adaptive sampling based on a luminance [SAB19], which is evaluated by the flux in RSMs. Given a luminance map, wavelet importance sampling [CJAMJ05] can be applied to rearrange the sampling points such that a region with high luminance is sampled more frequently. For the denoiser, if the computational cost of SVGF is not a concern, it can be improved to better accumulate photons instead. A-SVGF [SPD18] is a direct successor of the original SVGF, which adaptively reuses previous samples to adjust its temporal accumulation weight, leading to better temporal stability. Otherwise, a faster variant should be investigated instead, which might lead to a degradation of the quality inevitably. Finally, the RSMs and G-Buffer need to store many geometry attributes such as normals and material attributes, which leads to a massive memory footprint in high-resolution screens. A possible solution is to use a Visibility Buffer [BH13] instead, which stores only a primitive ID and a triangle ID. Due to the IDs locality, this guarantees a cache hit upon retrieving geometry attributes corresponding to those IDs, leading to much-improved memory efficiency.

6. Conclusion

We present a new approach to render ray-traced caustics in real-time, which derives Image-Space Ray Tracing to trace photons from one view to another view, and uses a denoiser to eliminate noise caused by Monte Carlo sampling. It does not require hardware-accelerated ray tracing and genuinely runs on every GPU. The quality of the caustics produced by our approach is comparable to the physically-based Path Tracing method. Moreover, our approach is scalable by the controllable number of photons for higher quality, at the cost of higher execution time.

Acknowledgments

Our source code utilizes AMD Radeon Cauldron framework [GPU21] as a foundation to further develop our methodology, as well as its Cauldron-Media library that contains the Sponza scene. Also, we would like to thank all of the anonymous reviewers for the feedback that helps us leveraging the paper's quality.

References

[BH13] BURNS C. A., HUNT W. A.: The visibility buffer: A cache-friendly approach to deferred shading. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (2013), 55–69. URL: http://jcgt.org/published/0002/02/04/. 7

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: Efficiently evaluating products of complex functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 24*, 3 (Aug. 2005), 1166–1175. doi:10/c79g6q. 7

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (2005), I3D '05, pp. 203–231. doi:10.1145/1053427.1053460. 2

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), I3D '06, Association for Computing Machinery, p. 93–100. doi:10.1145/1111411.1111428. 2

[EAMJ05] ERNST M., AKENINE-MÖLLER T., JENSEN H. W.: Interactive rendering of caustics using interpolated warped volumes. In *Proceedings of Graphics Interface 2005* (2005), pp. 87–96. 2, 4

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 192:1–192:10. doi:10.1145/2366145.2366211. 1

[GPU21] GPUOPEN: Radeon cauldron - the easy, extensible framework for experimenting in directx12 and vulkan, 2021. v. 1.3.2. URL: https://gpuopen.com/cauldron-framework/. 7

[Hut21] HUTTER M.: gltf materials, 2021. URL: https://github.khronos.org/glTF-Tutorials/gltfTutorial/gltfTutorial_010_Materials.html. 3

[Int21] INTEL: Intel® open image denoise: High-performance denoising library for ray tracing, 2021. v. 1.4.0. URL: https://www.openimagedenoise.org. 5

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques 96* (1996), Pueyo X., Schröder P., (Eds.), Springer-Verlag, pp. 21–30. 1

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, Association for Computing Machinery, p. 143–150. doi:10.1145/15922.15902. 5

[Kar14] KARIS B.: High-quality temporal supersampling. In *SIGGRAPH Courses: Advances in Real-Time Rendering in Games* (2014). 3

[KD13] KAPLANYAN A. S., DACHSBACHER C.: Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proc. of Eurographics 2013) 32*, 2 (2013), 63–72. 1

[Kim19] KIM H.: Caustics using screen-space photon mapping. In *Ray Tracing Gems*. Springer, 2019, pp. 543–555. 2, 4

[Kli08] KLINT J.: Deferred rendering in leadwerks engine, 2008. URL: http://www.leadwerks.com/files/Deferred_Rendering_in_Leadwerks_Engine.pdf. 3, 4

[Lux21] LUXCORERENDER: Luxcorerender - open source physically based renderer, 2021. v. 2.5. URL: https://luxcorerender.org. 5

[ML09] MCGUIRE M., LUEBKE D.: Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the ACM SIGGRAPH/EuroGraphics conference on High Performance Graphics 2009* (New York, NY, USA, August 2009), ACM. URL: https://casual-effects.com/research/McGuire2009Photon/index.html. 2

[MLM13] MARA M., LUEBKE D., MCGUIRE M.: Toward practical real-time photon mapping: efficient gpu density estimation. In *Proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games* (2013), pp. 71–78. 5

[MM14] MCGUIRE M., MARA M.: Efficient GPU screen-space ray tracing. *Journal of Computer Graphics Techniques (JCGT) 3*, 4 (December 2014), 73–85. URL: http://jcgt.org/published/0003/04/04/. 2, 3, 4, 7

[NW09] NICHOLS G., WYMAN C.: Multiresolution splatting for indirect illumination. I3D '09, Association for Computing Machinery, p. 83–90. doi:10.1145/1507149.1507162. 2

[NW10] NICHOLS G., WYMAN C.: Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Transactions on Visualization & Computer Graphics 16*, 05 (sep 2010), 729–741. doi:10.1109/TVCG.2009.97. 2

[SAB19] SMAL N., AIZENSHTEIN M., BENCHMARKS U.: Real-time global illumination with photon mapping. Springer, 2019, pp. 409–436. 2, 7

[SKP07] SHAH M. A., KONTTINEN J., PATTANAIK S.: Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 272–280. doi:10.1109/TVCG.2007.32. 2, 4, 5

[SKS11] SOUSA T., KASYAN N., SCHULZ N.: Secrets of cryengine 3 graphics technology. In *SIGGRAPH Courses: Advances in Real-Time Rendering in 3D Graphics and Games* (New York, NY, USA, 2011), ACM. 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. 1–12. 5

[Sou05] SOUSA T.: Generic refraction simulation. In *GPU Gems 2*. Addison-Wesley Professional, 2005, pp. 295–306. 2

[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), 1–16. 7

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), pp. 197–206. 2

[Sta96] STAM J.: Random caustics: natural textures and wave theory revisited. In *Proceedings of SIGGRAPH '96* (1996), p. 150. 2

[Sta15] STACHOWIAK T.: Stochastic screen-space reflections. In *SIGGRAPH Courses: Advances in Real-Time Rendering in Games* (2015). 2, 3, 4, 7

[WD06] WYMAN C., DAVIS S.: Interactive image-space techniques for approximating caustics. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), I3D '06, Association for Computing Machinery, p. 153–160. doi:10.1145/1111411.1111439. 2, 4

[WN09] WYMAN C., NICHOLS G.: Adaptive caustic maps using deferred shading. *Computer Graphics Forum 28*, 2 (2009), 309–318. doi:10.1111/j.1467-8659.2009.01370.x. 2, 4

[YK09] YUKSEL C., KEYSER J.: Fast real-time caustics from height fields. *The Visual Computer (Proceedings of CGI 2009) 25*, 5-7 (2009), 559–564. doi:10.1007/s00371-009-0350-4. 2, 4

[ZGJ20] ZELTNER T., GEORGIEV I., JAKOB W.: Specular manifold sampling for rendering high-frequency caustics and glints. *Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (July 2020). doi:10.1145/3386569.3392408. 1