EUROGRAPHICS 2009 / P. Dutré and M. Stamminger
(Guest Editors)

Volume 28 (2009), Number 2

Adaptive Caustic Maps Using Deferred Shading

Chris Wyman† and Greg Nichols

University of Iowa, USA

Abstract

Caustic maps *provide an interactive image-space method to render* caustics, *the focusing of light via reflection and refraction. Unfortunately, caustic mapping suffers problems similar to shadow mapping: aliasing from poor sampling and map projection as well as temporal incoherency from frame-to-frame sampling variations. To reduce these problems, researchers have suggested methods ranging from caustic blurring to building a multiresolution caustic map. Yet these all require a fixed photon sampling, precluding the use of importance-based photon densities. This paper introduces* adaptive caustic maps. *Instead of densely sampling photons via a rasterization pass, we adaptively emit photons using a deferred shading pass. We describe deferred rendering for refractive surfaces, which speeds rendering of refractive geometry up to 25% and with adaptive sampling speeds caustic rendering up to 200%. These benefits are particularly noticable for complex geometry or using millions of photons. While developed for a GPU rasterizer, adaptive caustic map creation can be performed by any renderer that individually traces photons, e.g., a GPU ray tracer.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Graphics researchers have long studied light-material interactions, often focusing on special cases such as perfectly diffuse and specular surfaces. Even the earliest path tracing papers [Kaj86] demonstrated caustics from reflective and refractive objects. Unfortunately, computational costs prohibit real time use of path tracing, and many fast global illumination algorithms restrict scenes to diffuse materials.

While non-diffuse materials have traditionally been difficult for interactive rasterization, a number of recent advancements using image-space approximations [OB07, Wym05], object-space approximations [EMDT06, RH06], and ray-based techniques [KBW06, SZS*08] allow applications to quickly incorporate simple reflections and refractions, though fully accurate renderings generally remain too costly.

Building on this work a number of researchers simultaneously developed caustic mapping [HK07, SKP07, SKALP05, WD06], which uses a two-pass process similar to photon mapping [Jen01]. Instead of creating a point cloud of photons hits, however, caustic mapping generates a caustic intensity map for projection onto the scene geometry in conjunction with a shadow map. While this enables interactive caustic rendering, common rendering problems remain:

1. Poor photon sampling due to rasterization on a regular grid leads to both under- and over-sampling.
2. Millions of photons are needed for sharp, high-quality

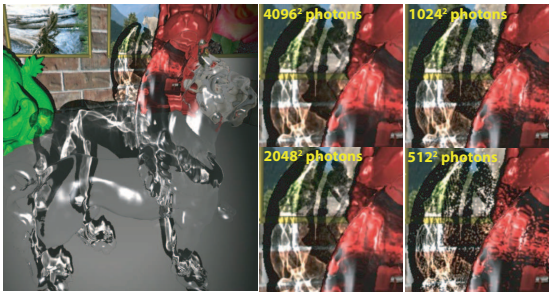

Figure 1: *Adaptively computed caustics from a 100k triangle feline model. Our results allow dynamic lighting, viewpoint, and geometry at 24 fps for quality equivalent to a* 4096^2 *regularly sampled photon grid. Lower quality results, for* 2048^2, 1024^2, *and* 512^2 *photon grids run at 50, 62, and 65 fps. The insets show varying quality settings.*

† E-mail: { cwyman | gbnichol }@cs.uiowa.edu

caustics; naive implementations must process each individually, which becomes prohibitively expensive.
3. Photon sampling locations can change between frames, leading to coherency problems such as popping artifacts.

Researchers proposed reducing noise and coherency artifacts via spatial or temporal blurring, spreading photon intensity over "caustic triangles," and dynamically changing a photon's area of influence based upon the amount of surface distortion. Other work uses hierarchical techniques to avoid processing every photon, especially those that do not interact with specular surfaces. Unfortunately, none of these techniques allow truly dynamic photon sampling. Photons hitpoints are first computed at some fixed sampling rate and either processed, discarded after processing determines photon irrelevancy, or ignored en masse via hierarchical processing.

This paper proposes a novel technique for adaptively sampling photons, allowing dynamic quality control for applications that must run on hardware with varying computational power and enabling different quality heuristics and error metrics. To permit adaptive processing, we first introduce a simple deferred shading technique that allows approximate refractions to be computed at individual pixels, rather than via more traditional rasterization (e.g., [SKALP05, Wym05]). Deferred shading not only enables adaptive caustics, it also speeds image-space refraction and allows approximate refraction from more than two interfaces. But our key improvement over hierarchical caustics techniques [Wym08] is the elimination of a fixed size photon buffer. Hierarchical caustic maps avoid *processing* unnecessary photons, we avoid *creating* them in the first place.

2. Previous Work

An enormous variety of research has sought to accurately and efficiently render caustics. Some work, such as path tracing [Kaj86], enables a comprehensive set of material and illumination effects without focusing on specific effects like caustics. Wavefront techniques model illumination by tracing waves through the scene [MH92], and while this approach has been proposed for both surface and volumetric caustics [IZT*07] it remains computationally expensive.

Reducing illumination costs in unimportant regions significantly speeds rendering. For general global illumination, techniques such as hierarchical radiosity [SAG94] use this approach. For caustics, Suykens [SW00] proposed distributing a photon's energy among neighbors in densely sampled areas. Unfortunately this requires identifying photons to distribute energy among, something caustic mapping seeks to avoid. A number of offline ray-based techniques adaptively sample photons, often using a small set of photons to improve estimates for additional samples [BAJ08, TJ97].

Recent work on interactive caustics emits beams [Wat90] or photons [Arv86, Jen01] from the light, tracing them backwards until an opaque surface is encountered and accumulating light intensity on these surfaces. Watt [Wat90] introduced caustic volumes that bound light beams as they distort via reflection and refraction. These volumes can be rendered using hardware acceleration [IDN02], though some care is required to handle non-linearities along the boundaries [EAMJ05].

Interactive photon tracing techniques work similar to the two-pass photon mapping process [Jen01]. Photon mapping uses a kD-tree during final rendering to accelerate final gathering from stored photons. As kD-tree builds on GPUs are difficult, researchers have tried avoiding the $O(n \log n)$ build cost using a simple 3D grid [PDC*03], but that leads to lengthier, non-interactive render times. Recent research has enabled dynamic kD-tree creation [ZHWG08] and traversal [FS05] using a GPU, leading to accurate, dynamic caustic rendering at around 10 frames per second for simple scenes.

2.1. Caustic Mapping

Creating a caustic map, a 2D illumination texture projected onto the scene similar to a shadow map, provides another way to avoid kD-tree construction. Caustic mapping uses three passes (see Figure 2): photon emission, rearrangement into the caustic map, and caustic map projection. Generally, photon emission rasterizes from the light's view to generate a regularly-sampled grid of photons. However, other techniques such as ray tracing [PBMH02] or vertex tracing [SR00] could replace this step.

Photon emission creates a *photon buffer*, a 2D image storing final photon hitpoints for all rays that hit specular surfaces. Instead of building an acceleration structure on these positions, photons are treated as geometry and drawn into a caustic map. Each map texel accumulates caustic intensity on the opaque surfaces closest to the light. This map works in conjunction with a shadow map; a shadow map allows quick lookups to determine direct lighting, whereas a caustic map allows quick lookups to identify indirect lighting from caustics. These lookups occur in the final render pass.

Typically the second step, caustic map creation, controls lighting quality and cost. Using more photons dramatically improves quality, but splatting each one into the caustic map quickly becomes the bottleneck. Szirmay-Kalos et al. [SKALP05] and Wyman and Davis [WD06] experimented with different photon counts, but because quality does not improve linearly as sampling increases, the millions of photons needed to interactively generate crisp, noise-free results remained infeasible.

Wyman and Dachsbacher [WD08] improved quality by varying splat size based on photon convergence or divergence. Umenhoffer et al. [UPSK08] proposed using caustic triangles instead of splats to reduce noise. Unfortunately both still require processing every photon, despite increased fidelity. Hierarchical caustic maps [Wym08] discard unimportant parts of the photon buffer en masse, enabling cheaper

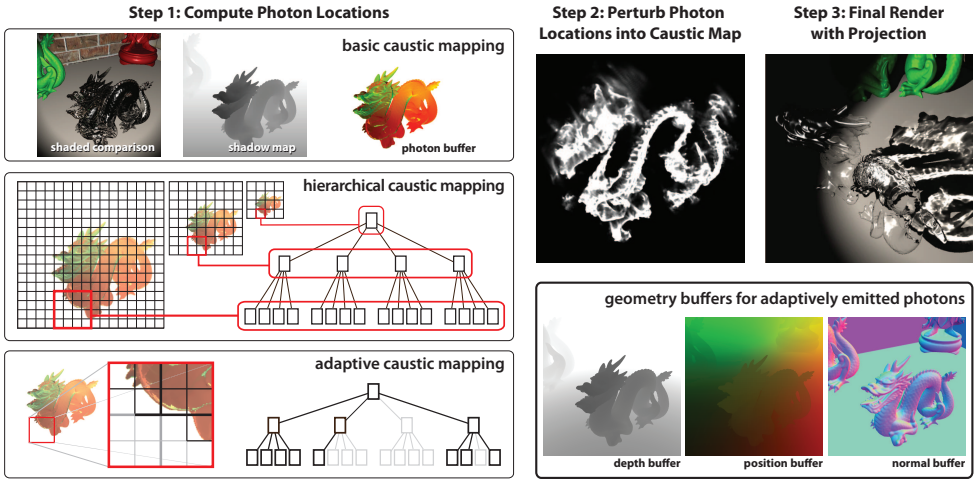

Figure 2: *Basic caustic mapping occurs in three steps. The first step creates a photon buffer storing the final hitpoints for photons interacting with specular surfaces; we provide a shaded light view and shadow map for comparison. The second step rearranges these photons by splatting into a caustic map. Finally, the caustic map is projected onto the scene. Hierarchical caustic maps improve performance by creating a mipmap-like hierarchy and processing in a top-down manner that avoids processing the entire photon buffer. We introduce a new, adaptive approach that never creates an explicit photon buffer. Instead, an adaptive deferred shading pass that point-samples the geometry buffers allows us to emit photons adaptively. This not only avoids processing the entire photon buffer, it never generates unused photons.*

processing of oversampled regions and reducing the cost of splatting using a multi-resolution caustic map.

3. Adaptive Caustics via Deferred Shading

Unlike shadow maps, where computing the ideal sampling rate before rendering is feasible [LGQ*08], arbitrary photon convergence and divergence forces developers to guess good sampling rates for fixed-sample caustic renderings. This leads to either oversampled or undersampled caustics at any distance from the viewer. Ideally, applications would adaptively choose how many photons to shoot. Hierarchical caustic maps (HCMs) [Wym08] approach this ideal, but a fixed maximal sampling resolution must be chosen *a priori* and photons are emitted at this resolution. Caustic map creation then requires a hardware-accelerated mipmap build followed by a traversal of the mipmap quadtree to avoid processing extraneous photons.

To clarify the problem, previous caustic mapping techniques fix the number of photons and their sampling locations prior to emitting the photons. We propose emitting a few photons, and adaptively refining with additional photons until the desired quality is attained. Algorithmically this means that instead of first creating a photon buffer and then processing it to generate a caustic map, these two steps become coupled. If we notice adjacent photons from the photon buffer converge in the caustic map, we need not refine the area with more photons; if neighbors diverge, additional photons are emitted to help reduce noise.

Existing caustic mapping approaches "emit" photons by rasterizing the specular object from the light's point of view and treating each fragment as a photon. Thus, controllable hierarchical rasterization (e.g., [Gre96]) would neatly allow adaptive photon emission. Unfortunately, fixed-function rasterization hardware makes this infeasible on current GPUs.

Instead, we observe that deferred shading [ST90] also avoids extraneous processing, postponing final illumination computations until visible fragments are identified. Generally, the final deferred shading pass is instantiated by drawing a full-screen quad. We propose a slightly different deferred shading approach. Instead of triggering the deferred pass with a full-screen quad (to simultaneously emit all photons), we will point sample the geometry buffers during our adaptive photon emission; if we do not need a photon, we never sample it. This contrasts with hierarchical caustic mapping [Wym08], which always generates a dense, regularly sampled grid of photons and then skips processing irrelevant ones. Our adaptive technique simply *never generates* these irrelevant photons.

3.1. Deferred Shading for Refraction

Before describing our deferred photon sampling, we first examine a simpler problem: how to use deferred shading for rendering specular materials. Because researchers have proposed varying incompatible techniques for rendering reflective and refractive geometry, describing a general deferred renderer is beyond the scope of this paper. Instead, we explore a deferred approach for image-space refraction [Wym05] and assert that with simple modifications this applies to a subset of other techniques.

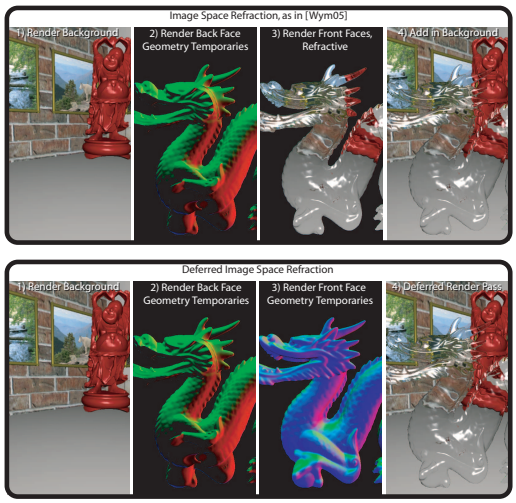

Figure 3: *Compare image-space refraction and our deferred rendering approach. One key difference is that steps 2 and 3 for deferred rendering can be combined, touching refractor geometry to output front and back faces just once.*

The original image-space refraction approximation requires four passes (see Figure 3). First, opaque geometry behind the refractor is rendered, storing color and depth. A second pass stores surface normals and depths for the backside of the refractor. The third step rasterizes the refractor, approximating a doubly refracted ray at each fragment. Finally, the refractor is combined with the opaque geometry. Note that the refractor may have a depth complexity greater than two, causing extraneous shader executions to occur for some pixels. This problem is commonly addressed by deferred shading: storing geometry buffers during a cheap render pass and executing final shading exactly once per pixel by rendering a full screen quad.

While image-space refraction already stores geometry buffers, these contain geometry only for the back-facing geometry. Thus we reformulate the refraction passes to store geometry buffers containing normals and depth for *both* front and back refractor surfaces:

1. Render color and depth of geometry behind the refractor.
2. Render back of refractor, storing normals and depth.
3. Render front of refractor, storing normals and depth.
4. Render a full screen quad, approximating refraction if the pixel lies on refractor else copying color from step 1.

This deferred process is beneficial in two ways. Hidden fragments are never shaded. We found overdraw on complex refractive objects as high as 10%, even with culling enabled. Secondly, steps 2 and 3 can be combined into a single pass that touches refractor geometry only once; a geometry shader selectively outputs front surface data to one buffer and back surface data to another. Prior work required rendering refractor geometry twice to achieve refraction.

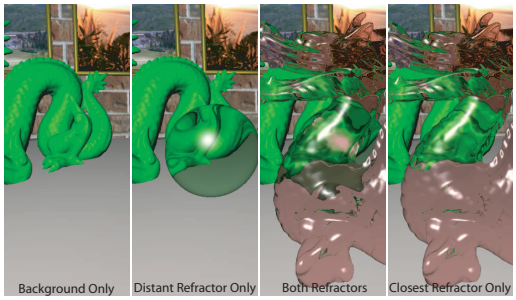

Figure 4: *Multi-layer refraction approximated with back-to-front composition of deferred refraction passes. From left to right: the opaque background, deferred refraction of a distant sphere, a refractive dragon with the sphere visible behind, and (for comparison) the dragon without the sphere.*

3.1.1. Multi-Layer Deferred Refraction

Beyond speed improvements from eliminating overdraw and processing refractive geometry only once, deferred shading enables plausible rendering of one refractor seen through another. Given a coarse sorting of refractive objects, they are processed via deferred shading from back to front. For two refractors (see Figure 4):

1. Render background,
2. Render furthest refractor's geometry buffers,
3. Render full screen quad to display furthest refractor,
4. Render closer refractor's geometry buffers, and
5. Render full screen quad to display final result, using result from step 3 as the "background."

Note that the refraction angle is incorrect at interfaces beyond the 2nd. Since step 3 does not know the correct direction of any incident ray from step 5, the refractor is treated as if viewed from the eye. This behaves somewhat similar to the approximation of Kay and Greenberg [KG79] in the presence of multiple refractors. Also note this maintains all other limitations of image-space refraction [Wym05], including the inability to bend rays around background objects.

3.2. Deferred Shading for Caustic Rendering

Traditional deferred renderers shade all pixels in parallel, drawing a screen-sized quad that executes a shader only once per pixel. Because all required information has been precomputed, pixels can theoretically be shaded in any order. This observation allows us to create a photon buffer adaptively, using an initial coarse sampling and refining where needed, instead of rendering a complete photon buffer and processing it hierarchically. Not only does this avoid processing most irrelevant photons, it avoids *creating* them in the first place.

As discussed in Section 3.1, prior refraction algorithms and prior caustic mapping algorithms approximated refraction by rasterizing the refractive geometry. Due to GPU re-

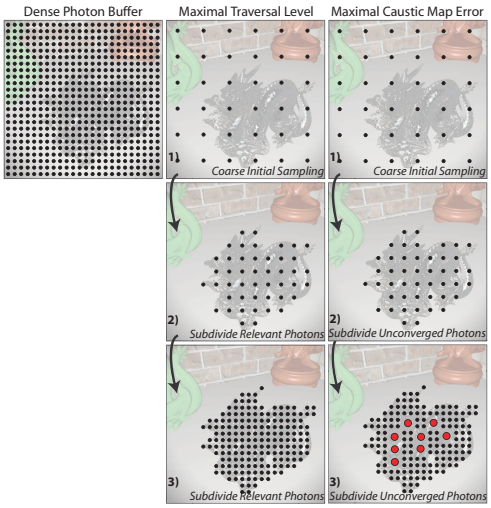

Figure 5: *Compare previous dense photon sampling with adaptive sampling using our proposed metrics. For the* maximal traversal level*, relevant photons are subdivided until reaching some finest subdivision level. The* maximal caustic map error *subdivides relevant photons unless the children converge, when a single photon cluster is emitted. Photon clusters are not considered during further subdivisions.*

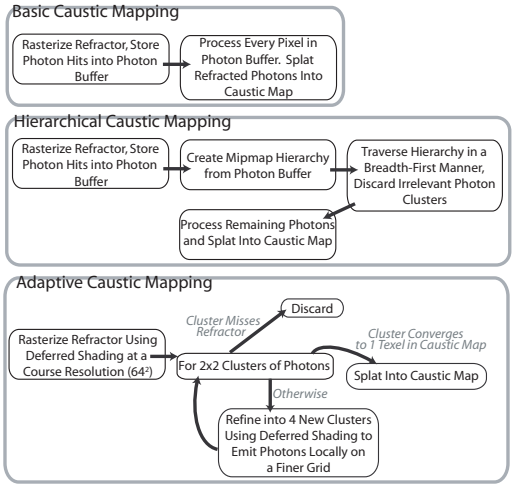

Figure 6: *A high-level comparison of caustic map creation via basic, hierarchical, and adaptive caustic mapping.*

quirements, rasterization outputs into a fixed-size image. In eye space, this fixed-size image is the final rendering. In light space, this fixed-size image is the photon buffer.

We propose to get around this limitation using adaptive deferred shading (see Figure 5), which starts by coarsely generating photons using deferred shading. We used a 64^2 grid for this course sampling. For each 2×2 cluster of photons, we discard the photons if they miss the refractor, we splat them into the caustic map if our termination criteria is met, otherwise we refine each cluster into four new clusters by generating new photons on a finer grid using deferred shading.

At a high level, caustic mapping changes to:

1. Render geometry buffers for deferred photon creation,
2. Traverse a virtual photon hierarchy, lazily create necessary photons via deferred rendering, and output a list of relevant photons at their correct resolution,
3. Splat these photons into a caustic map,
4. Render from eye, projecting caustic map onto scene.

Figure 6 compares this approach with basic and hierarchical caustic mapping, with the key change being the avoidance of an explicit photon buffer, which inevitably contains irrelevant photons.

However, two issues must be still dealt with: what resolution to render geometry buffers, and how to stop traversal of the adaptive photon hierarchy. Ideally, geometry buffers for deferred shading would also be dynamically created. Unfortunately, without ray tracing or a hierarchical rasterizer that exposes intermediate steps, this data is difficult to dynamically generate. We observe that unlike a photon buffer, which stores photon positions that vary in complex and nonlinear ways, geometry buffers store object positions and normals. Interpolating these quantities is straightforward, allowing dynamic sampling at arbitrary locations in exchange for smoothing caustic variations due to small, missing geometric detail. We found that sampling the refractor positions and normals at a quarter the final caustic map's resolution provided a good speed-quality tradeoff.

We implemented two photon refinement metrics to identify when to stop photon refinement. Sections 3.2.1 and 3.2.2 introduce the *maximal traversal level* and *maximal caustic map error* metrics, and Figure 5 compares the photons generated with these metrics to prior dense photon samplings.

3.2.1. Metric: Maximal Traversal Level

The *maximal traversal* metric simply continues generating denser photon sampling, using regular sampling on a finer grid, until one of two conditions are reached:

1. If all photons in the current 2×2 cluster miss the refractor, none are output.
2. When sampling has reached some maximal subdivision level, all remaining photons are output.

This gives comparable results to a basic caustic map, which renders a maximal resolution photon buffer and uses all photons that intersect the refractor.

3.2.2. Metric: Maximal Caustic Map Error

The *maximal caustic map error* metric works similar to the maximal traversal metric, generating denser regular photon samples, until one of two conditions are reached:

1. If all photons in the current 2×2 cluster miss the refractor, none are output.
2. When all photons in the cluster converge to a single caustic map texel, one photon is output with intensity based upon the solid angle of all cluster photons.

Ideally, this continues until no noise is visible in the caustic map. Unfortunately, due to arbitrary divergence of specularly reflected and refracted photons, this error metric requires an arbitrary number of photons.

We found that after traversing 14 levels in the hierarchy, equivalent to sampling on a 16384^2 regular-grid, a significant percentage of divergent photons remained well above the error threshold. This generates enormous lists of photons to splat into the photon buffer, often exceeding the memory available on current graphics accelerators. Thus, our implementation adds a third condition. Once a specified maximal traversal level is reached all photons are emitted. Alternatively, photon divergence greater than a single texel but less than a user-defined tolerance could terminate traversal. Remaining noise is eliminated by rendering to a multiresolution caustic map, splatting into lower resolution maps for diverging photons [WD08].

4. Implementation

Our prototype uses OpenGL with the transform feedback and geometry shader extensions. Characteristics of the GPU stream processing model thus affect numerous design choices as well as performance. In particular, error metric implementation, poor parallelism during early traversal steps, and high memory consumption for photon storage provided challenges.

4.1. Error Metric Implementation

While the maximal traversal metric generates significantly more photons than the maximal error metric (see Figure 7), it still runs faster until an adaptively generated photon buffer larger than 8192^2 is used. This anomaly occurs because the maximal traversal metric requires only a single stream kernel per traversal step, as each input photon either gets discarded or subdivided into four children. For the maximal error metric, each traversal step requires *three* kernels: one that computes photon hit positions, one that identifies converged photons and outputs a separate stream, and one that identifies unconverged photons and subdivides them. Ideally this requires only two kernels, but without the third pass our prototype required additional intermediate buffers that increased memory requirements, decreased memory coherency, and reduced speed and scalability.

4.2. Avoiding Serial and Extraneous Processing

To maintain parallelism during photon generation, we do not start our adaptive traversal with a single photon. Instead we begin traversal with enough photons to keep all units on the GPU busy (a 64^2 regularly sampled grid). Also, we always use the maximal traversal metric for the first traversal steps, since coarsely sampled photons do not generally converge to a single caustic map texel. Checking convergence in early traversal steps adds significant overhead for little benefit. We begin checking for convergence once we reach the 512^2 subdivision level.

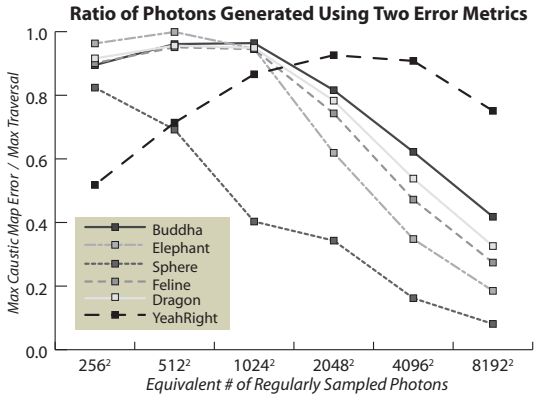

Figure 7: *The* maximal caustic map error *metric significantly reduces the total number of photons when compared to the* maximal traversal level *metric. This graph gives a ratio of the photons created with these metrics at various maximal traversal levels.*

4.3. Lowering Memory Usage With Photon Batching

While our adaptive technique generates only relevant photons from a dense grid, avoiding storage of an extremely large floating-point photon buffer, we still produce lots of data. A full 8192^2 photon buffer requires 512 MB of video memory, but with just relevant photons (around 10% of photons in our scenes) memory requirements still top 50 MB. Figure 7 shows that intelligent traversal metrics can reduce data output by 20–60%, though our implementation uses additional buffers that nullify this memory reduction.

Fortunately, deferred shading allows computations in any order. Our initial adaptive pipeline descends the photon tree in a breadth-first manner, generating all photons prior to splatting into the caustic map. To avoid storing all photons, we can split photons into *batches*. We adaptively generate photons breadth-first until we exceed a user-defined memory limit. We then split photons into independent batches. Batches are processed one at a time, refining photons and splatting them into the caustic map before continuing.

This allows nearly arbitrary reduction in memory usage, though additional passes add overhead. Figure 8 demonstrates the overhead introduced by changing the number of batches. We found two to sixteen additional batches slow caustic map creation by 25–300% (corresponding to a 5–25% reduction in framerate), but it allows memory reduc-

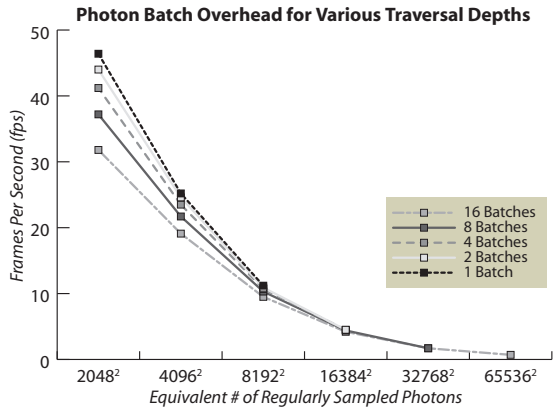

Figure 8: *Overhead from multiple photon batches for the Buddha scene. Without photon batching, traversing 14 levels (16384^2) in the photon hierarchy was impossible.*

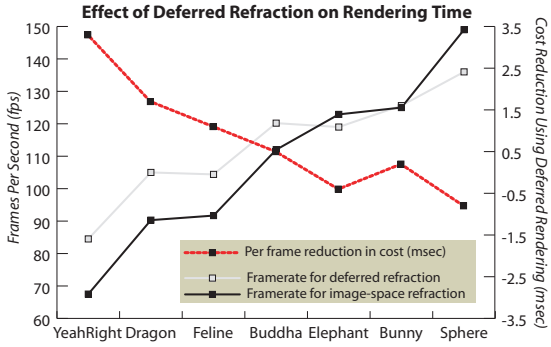

Figure 9: *Graph comparing costs of previous image-space refraction and deferred shading. Performance on low-polygon models actually degrades, as repeatedly rendering these models costs less than the overhead to defer shading.*

tion sufficient to use a grid of 65536^2 photons. A full 65536^2 photon buffer requires 32 GB; even our non-batched adaptive traversal requires 250 MB of photon storage (plus an additional 500 MB in temporaries). Splitting photons into 16 batches, memory requirements are reduced to 16 MB for photon storage and an additional 32 MB of temporary space.

5. Results and Discussion

Results presented below were benchmarked on a quad-core Intel Xeon processor at 2.6 GHz with a GeForce GTX 280. All timings use a final output resolution of 2048^2, downsampled to a 1024^2 window for an antialiased rendering. Results of our adaptive technique are compared to densely sampled caustic maps with an "equivalent" number of photons. This reflects the maximal traversal level we descend to during photon generation, so caustic crispness and noise will be comparable to a that from a similar resolution dense photon buffer.

Figure 9 compares rendering speeds for refractive geometry using image-space refraction [Wym05] and our deferred approach from Section 3.1. Generally, deferred rendering speeds refraction by 5–25% on moderate sized models. However, low polygon objects without significant overdraw can perform up to 10% worse with deferred rendering. The deferred pass adds an additional temporary buffer, and for simple objects the overhead for this buffer negates any savings. To clarify data collection and reporting, however, all other timings use deferred shading for final rendering.

Figure 10 shows the effects of deferred caustic map creation and of adaptive photon generation using the metrics from Sections 3.2.1 and 3.2.2. As with deferred refraction, complex models exhibit more dramatic speedups and simple objects perform worse. As the number of emitted photons increases, the difference between deferred and non-deferred rendering shrinks as other steps become the bottleneck.

Photon Buffer Res.	Photons Processed With Various Rendering Methods: Caustic Maps	Hier. Caustic Maps	Adaptive with Max Traversal	Adaptive with Max Map Error
256^2	65,656	4,803	5,092	4,556
512^2	262,144	19,861	20,160	19,380
1024^2	1,048,576	78,538	81,268	78,320
2048^2	4,194,304	286,298	326,896	266,768
4096^2	16,777,216	947,559	1,312,524	816,888
8192^2	67,108,864	N/A	5,210,492	2,182,704
16384^2	268,435,456	N/A	12,582,912	5,398,900

Table 1: *Comparison of photons splatted into the caustic map (for the Buddha scene) using basic caustic maps, hierarchical caustic maps, and our adaptive caustic maps using both the error metrics. Note that previous techniques could not handle more than* 4096^2 *photons.*

Adaptive photon generation incurs additional overhead, as two to eight traversal steps are needed for 256^2 to 16348^2 photons. As discussed in Section 4.1, traversal with the caustic map error metric is more costly. For extremely complex objects, such as the 755k triangle YeahRight model, traversal overhead is comparatively small and adaptive caustic mapping always proves advantageous (see Figure 10). For other objects, adaptive caustic mapping runs faster with at least 1024^2 photons. At this level a dense photon buffer contains 1 million photons (see Table 1), and the savings by creating less than 10% overcomes the traversal overhead. Using 2048^2 photons, adaptive sampling is 15% faster even with a simple sphere and complex models run twice as fast. With finer sampling the number of photons processed becomes the bottleneck, so refractor size becomes more important than geometric complexity.

Figure 11 shows per-frame costs to demonstrate the rendering bottleneck at various sampling densities. Notice that fixed costs rise roughly linearly with increased model complexity. Photon traversal costs instead vary with number of photons generated, which changes depending based on refractor size and light field of view. Interestingly, with our work generating a caustic map with 2048^2 photons costs roughly the same as rendering a 2048^2 image from the eye.

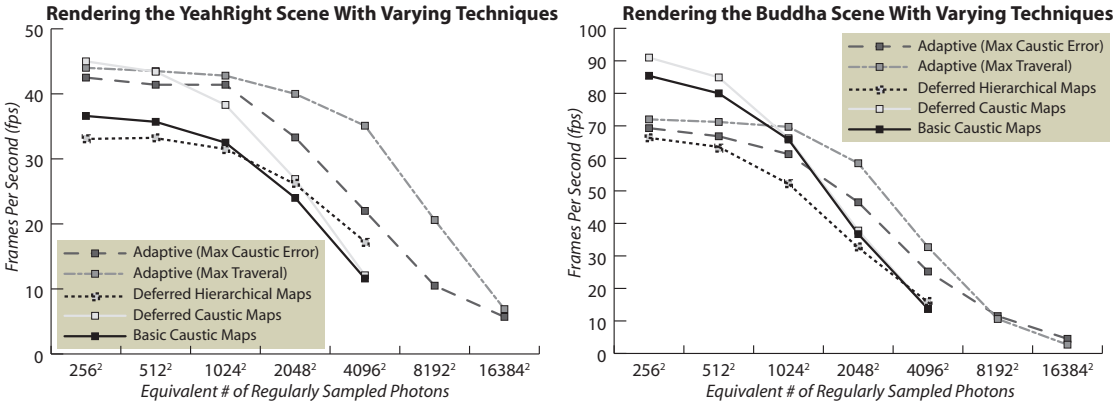

Figure 10: *Variations in rendering speed using five different caustic mapping techniques: simple caustic maps [WD06], deferred caustic maps (i.e., a dense photon buffer with a deferred render pass), deferred hierarchical caustic maps (based upon [Wym08]), and adaptive caustic maps using the maximal traversal and maximal caustic map error metrics.*

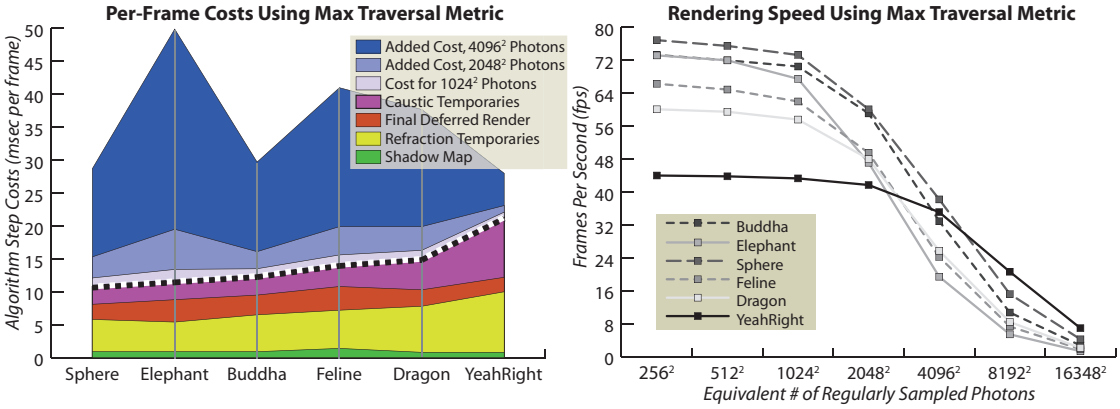

Figure 11: *The left chart breaks out per-frame costs of individual steps in our adaptive algorithm using the maximal traversal metric. Steps below the thick dashed line are fixed, no matter how many photons are adaptively generated. Costs reported using 4096^2 and 2048^2 photons are the incremental costs beyond the next coarsest sampling. The right graph demonstrates how final framerates for our scenes vary with changes to the maximal traversal level.*

Previous techniques could not handle more than 4096^2 photons, though our adaptive approach remains interactive with 8196^2 and even 16384^2 photons. Using more than 4096^2 photons requires batching the photons, which incurs the additional overhead shown in Figure 8. While our graphs stop at 16384^2, our approach scales to much finer photon sampling. Using the maximal caustic map error metric, the sphere scene converges with a 131072^2 photon buffer while still running at 2 fps. The Buddha scene converges with around 524288^2 photons, running at 0.3 fps.

Figures 1, 12, and 13 show results of our adaptive sampling. Because deferred refraction gives results identical to previous work, modulo antialiasing variations at silhouettes, only Figure 3 shows both approaches. Figure 13 shows an example of caustics from more than two refractive interfaces, using multi-layered deferred refraction when rendering photons. Because these layers are cheap and each refractive object is rasterized just once during photon emission,

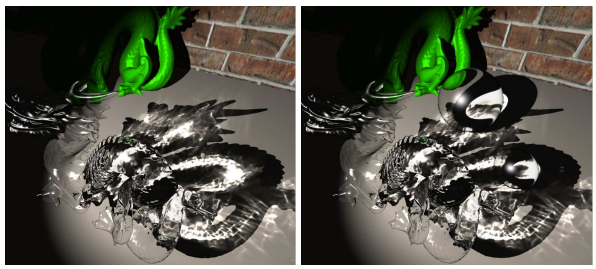

Figure 13: *Caustics from a refractive dragon. (Left) With just the dragon the scene runs at 35 fps. (Right) Adding two occluded spheres slows rendering to 34 fps.*

caustics from multi-layered refractions are not much slower than from a single object; the refractive dragon and spheres runs at 34 fps compared to 35 fps for the dragon alone.

Figure 14 compares our rendering of the YeahRight

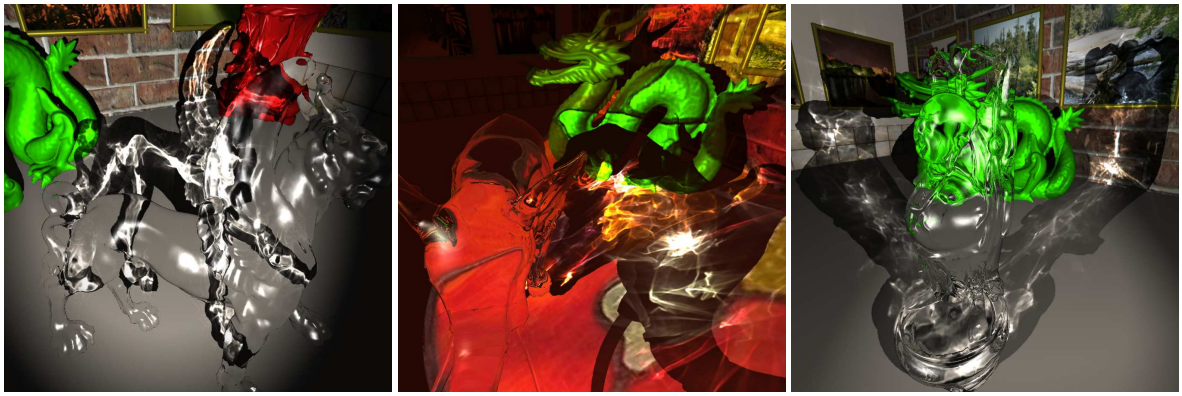

Figure 12: *Example scenes using adaptive caustic maps with the equivalent of* 4096^2 *photons per light. (Left) The feline model, (center) an elephant illuminated by a textured spotlight, and (right) a Buddha lit by two lights. Figure 11 provides timings for the feline and elephant. The Buddha runs at 26 fps or 17 fps using, respectively, one or two lights and the maximal error metric.*

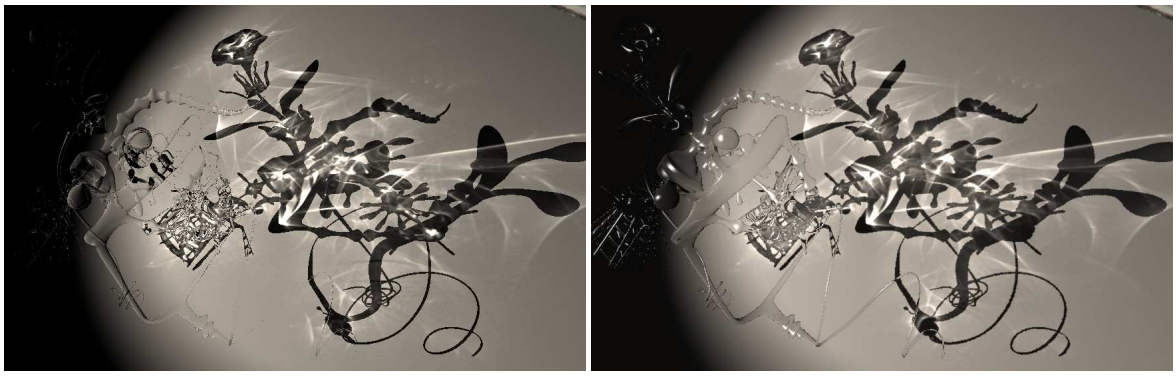

Figure 14: *(Left) Ground truth ray traced rendering of the "YeahRight" model using* 8196^2 *photons sampled on a regular grid. (Right) Interactive rendering at 21 fps using our adaptive technique and the same number of photons.*

model, running at 21 fps, to an offline ground truth image using the same photon sampling. Differences are largely due to two issues. The interactive rendering only approximates true refraction, so the refractor looks slightly different and photons that interact with the model multiple times are incorrect. Also, the ray traced comparison appears noisier in unconverged regions due to differing final gathers.

6. Conclusions

This paper introduced adaptive caustic mapping, a technique that uses deferred shading to generate caustic maps with only relevant photons and without rendering a largely ignored, dense photon buffer. We described a deferred shading approach for rendering refractive objects and demonstrated its applicability to plausible refractions through multiple objects and our adaptive caustic generation. Additionally, reordering the deferred shading computations allows generation of caustic photons in a variety of orders, corresponding to different adaptive traversal metrics or to create photon batches that reduce memory requirements.

These techniques provide numerous benefits:

- The ability to render high quality, crisp caustics while maintaining interactive or even realtime framerates.
- Developers need not specify how many photons to emit. Photon generation ends after achieving a desired accuracy.
- Photon count, rather than refractor complexity, controls performance. The light frustum need not tightly bound the refractor; except at the coarsest sampling, photons outside the refractor are never generated.
- Photons batches allow specified memory limits, rather than varying with the photon buffer size.
- Deferred shading avoids rasterizing geometry unnecessarily and allows rendering multi-layered refractions.
- Extraordinary numbers of photons may be simulated. Our prototype handles the equivalent of a 524288^2 regular sampling of photons while still responsive to user input.

While we believe adaptive caustic maps and deferred refraction provide many benefits, there are some limitations. Both methods add overhead that decreases performance on low polygon refractors or photon counts below 1024^2. Complex traversal metrics can only be crudely implemented with

current GPUs. The maximal error metric produces significantly fewer photons than a maximal traversal level yet still underperforms for most reasonable sampling rates. We showed our adaptive technique only on refractive caustics; with simple changes we could use other renderers that generate photons in any order (e.g., a GPU ray tracer).

Various future directions exist. Other traversal metrics, such as an eye-space error threshold, might improve speed, decrease noise, and reduce traversal depths. However, efficient metric implementation will prove vital. While we adaptively generate the photon buffer, adaptive subdivision of the caustic map would eliminate projection aliasing, similar to perspective shadow mapping techniques (e.g. [LGQ*08]). Finally, we believe adaptive photon generation would be ideal for volumetric caustics (e.g., [SZS*08]).

References

[Arv86] ARVO J.: Backward ray tracing. *Developments in Ray Tracing* (1986), 259–263. ACM SIGGRAPH Course Notes.

[BAJ08] BUDGE B., ANDERSON J., JOY K.: Caustic forecasting: Unbiased estimation of caustic lighting for global illumination. In *Proc. Pacific Graphics* (2008).

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

[EMDT06] ESTALELLA P., MARTIN I., DRETTAKIS G., TOST D.: A gpu-driven algorithm for accurate interactive reflections on curved objects. In *Proc. Eurographics Symp. on Rendering* (2006), pp. 313–318.

[FS05] FOLEY T., SUGERMAN J.: Kd-tree acceleration structures for a GPU raytracer. In *Proc. Graphics Hardware* (2005), pp. 15–22.

[Gre96] GREENE N.: Hierarchical polygon tiling with coverage masks. In *Proc. ACM SIGGRAPH* (1996), pp. 65–74.

[HK07] HU W., KAIHUAI Q.: Interactive approximate rendering of reflections, refractions, and caustics. *IEEE Trans. Vis. Comput. Graph. 13*, 1 (2007), 46–57.

[IDN02] IWASAKI K., DOBASHI Y., NISHITA T.: An efficient method for rendering underwater optical effects using graphics hardware. *Comput. Graph. Forum 21*, 4 (2002), 701–711.

[IZT*07] IHRKE I., ZIEGLER G., TEVS A., THEOBALT C., MAGNOR M., SEIDEL H.-P.: Eikonal rendering: efficient light transport in refractive objects. *ACM Trans. Graph. 26*, 3 (2007), Article 59.

[Jen01] JENSEN H. W.: *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[Kaj86] KAJIYA J.: The rendering equation. In *Proc. ACM SIGGRAPH* (1986), pp. 143–150.

[KBW06] KRUGER J., BURGER K., WESTERMANN R.: Interactive screen-space accurate photon tracing. In *Proc. Eurographics Symp. on Rendering* (2006), pp. 319–329.

[KG79] KAY D. S., GREENBERG D.: Transparency for computer synthesized images. In *Proc. ACM SIGGRAPH* (1979), pp. 158–164.

[LGQ*08] LLOYD B., GOVINDARAJU N., QUAMMEN C., MOLNAR S., MANOCHA D.: Logarthmic perspective shadow maps. *ACM Trans. Graph. 27*, 4 (2008).

[MH92] MITCHELL D., HANRAHAN P.: Illumination from curved reflectors. In *Proc. ACM SIGGRAPH* (1992), pp. 283–291.

[OB07] OLIVEIRA M. M., BRAUWERS M.: Real-time refraction through deformable objects. In *Proc. ACM Symp. on Interactive 3D Graphics* (2007), pp. 89–96.

[PBMH02] PURCELL T. J., BUCK I., MARK W. R., HANRAHAN P.: Ray tracing on programmable graphics hardware. *ACM Trans. Graph. 21*, 4 (2002), 703–712.

[PDC*03] PURCELL T., DONNER C., CAMMARANO M., JENSEN H. W., HANRAHAN P.: Photon mapping on programmable graphics hardware. In *Proc. Graphics Hardware* (2003), pp. 41–50.

[RH06] ROGER D., HOLZSCHUCH N.: Accurate specular reflections in real-time. *Comput. Graph. Forum 25*, 3 (2006), 293–302.

[SAG94] SMITS B., ARVO J., GREENBERG D.: A clustering algorithm for radiosity in complex environments. In *Proc. ACM SIGGRAPH* (1994), pp. 435–442.

[SKALP05] SZIRMAY-KALOS L., ASZODI B., LAZANYI I., PREMECZ M.: Approximate ray-tracing on the gpu with distance impostors. *Comput. Graph. Forum 24*, 3 (2005), 685–704.

[SKP07] SHAH M., KONTTINEN J., PATTANAIK S.: Caustics mapping: An image-space technique for real-time caustics. *IEEE Trans. Vis. Comput. Graph. 13*, 2 (2007), 272–280.

[SR00] STARK M., RIESENFELD R.: Exact illumination in polygonal environments using vertex tracing. In *Proc. Eurographics Rendering Workshop* (2000), pp. 149–160.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proc. ACM SIGGRAPH* (1990), pp. 197–206.

[SW00] SUYKENS F., WILLEMS Y.: Density control for photon maps. In *Proc. Eurographics Rendering Workshop* (2000), pp. 23–34.

[SZS*08] SUN X., ZHOU K., STOLLNITZ E., SHI J., GUO B.: Interactive relighting of dynamic refractive objects. *ACM Trans. Graph. 27*, 3 (2008), Article 35.

[TJ97] TAMSTORF R., JENSEN H. W.: Adaptive sampling and bias estimation in path tracing. In *Proc. Eurographics Rendering Workshop* (1997), pp. 285–295.

[UPSK08] UMENHOFFER T., PATOW G., SZIRMAY-KALOS L.: Caustic triangles on the gpu. In *Proc. of Computer Graphics International* (2008).

[Wat90] WATT M.: Light-water interaction using backward beam tracing. In *Proc. ACM SIGGRAPH* (1990), pp. 377–385.

[WD06] WYMAN C., DAVIS S.: Interactive image-space techniques for approximating caustics. In *Proc. ACM Symp. Interactive 3D Graphics* (2006), pp. 153–160.

[WD08] WYMAN C., DACHSBACHER C.: Improving image-space caustics via variable-sized splatting. *J. Graph. Tools 13*, 1 (2008), 1–17.

[Wym05] WYMAN C.: An approximate image-space approach for interactive refraction. *ACM Trans. Graph. 24*, 3 (2005), 1050–1053.

[Wym08] WYMAN C.: Hierarchical caustic maps. In *Proc. ACM Symp. Interactive 3D Graphics* (2008), pp. 163–171.

[ZHWG08] ZHOU K., HOU Q., WANG R., GUO B.: Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics (to appear)* (2008).