Interactive Caustics Using Local Precomputed Irradiance

Chris Wyman*
Department of Computer Science
The University of Iowa
Iowa City, IA

Charles Hansen
School of Computing
The University of Utah
Salt Lake City, UT

Peter Shirley
School of Computing
The University of Utah
Salt Lake City, UT

Abstract

Bright patterns of light focused via reflective or refractive objects onto matte surfaces are called "caustics". We present a method for rendering dynamic scenes with moving caustics at interactive rates. This technique requires some simplifying assumptions about caustic behavior allowing us to consider it a local spatial property which we sample in a pre-processing stage. Storing the caustic locally limits caustic rendering to a simple lookup. We examine a number of ways to represent this data, allowing us to trade between accuracy, storage, run time, and precomputation time.

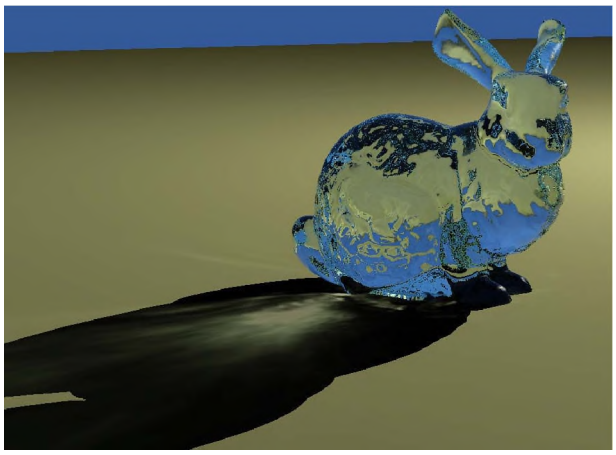

Figure 1. This scene runs at 2.3 fps with dynamic caustics and 2.6 fps without caustics, using a raytracer running on 30 CPUs for rendering.

1. Introduction

Daily life immerses us in environments rich in illumination we wish to capture in our renderings. Unfortunately rendering complex illumination often incurs a significant computational cost. Since many applications require interactive speeds, costly algorithms for global illumination are often infeasible.

Many applications could benefit from fast and simple algorithms for global illumination. Such algorithms exist and are used in various fields ranging from research to entertainment. These techniques vary in physical accuracy from exact radiosity solutions to fabricated lightmaps used to texture map surfaces in many of today's games. These methods typically suffer from a common computer graphics problem—poor scaling with scene complexity. Often techniques which run quickly on simple scenes bog down when used on a complex environment. Generally, much effort spent is computing illumination with only a minor impact on the image and negligible perceptual impact. In fact, while global illumination provides humans perceptual cues as to relative object locations, accuracy is not always important [12, 17].

While global illumination appears to have a significant impact upon how humans view interactions between objects, computing a full global illumination solution is often unnecessary. For example, sunlight reflected off a wooden pencil onto the wall across the room contributes little to a scene and can safely be ignored. While some researchers [26] have looked into simplifying the environment to reduce unnecessary computations, questions remain as to what level of simplification compromises the perceived quality of global illumination.

In this paper, we examine the focusing of light caused by reflective and refractive surfaces. This focusing, known as a "caustic," potentially affects the entire environment, yet in most cases appears in a relatively localized space around a specular object. One might see a caustic from a glass figurine on a table, for example, or the caustic from a mirror on an adjacent wall.

Our technique samples the caustic near the focusing ob-

* Work completed while at the University of Utah.

Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)
1550-4085/04 $ 20.00 IEEE

ject. This allows us to reduce caustic rendering from a global problem to a localized property which can be computed with a simple lookup. We perform this lookup at interactive framerates, even as objects and lights move. However, sampling takes significant precomputation and memory and accurate caustics are limited to the sampled region.

The rest of the paper is divided as follows. Section 2, outlines previous work in computing and speeding up global illumination. Section 3 discusses the behavior of caustics and shows how we deal with their complexities. Section 4 discusses various ways to sample a caustic and the tradeoffs involved and section 5 discusses some issues involved in rendering a caustic from sampled data. Section 6 presents our results. Finally, Section 7 presents our discussion, conclusions, and future work.

2. Background

As global illumination is important for many scenes, researchers have proposed many illumination models. Many existing techniques focus on diffuse interactions or do not handle all specular effects. We focus our attention in this section on techniques which generate caustics and interactive techniques similar to ours.

Accurate caustics have been generated semi-analytically for smooth surfaces [20], albeit at a cost too high for interactivity. Researchers have also investigated accurate interpolation between specular rays [2, 5], but these techniques have not yet yielded fast sampling-based methods for accurate caustic generation.

Pathtracing [16] generates beautiful global illumination renderings, but accuracy comes at extreme computational cost. Numerous researchers have looked into speeding up pathtracing and raytracing [18, 24, 29, 31]. These methods typically rely on storing previously computed samples and reprojecting them for a new viewpoint, sampling the places where errors are greatest. Unfortunately, in the case of moving caustics, the errors will be highest in the areas most expensive to recompute—the caustic.

Sending rays from the light has been successfully applied to generate caustics [1]. Many researchers have since used this technique, and it has been extended to include non-diffuse surfaces (e.g., photon mapping [14, 15]). This gives excellent caustics with much higher efficiency than pathtracing. While the results are view-independent, they require a reasonably expensive preprocess which must be repeated after moving a light or an object, and even recent work implementing photon mapping in hardware [25] is still far from interactive. Combinations of photon shooting and pathtracing have been examined [30]. By utilizing significant CPU resources, they could interactively render scenes with global illumination, including simple caustics. Such techniques can only shoot a limited number of photons per frame. Since higher quality caustics and caustics for complex objects require significant numbers of photons, such techniques cannot always quickly recompute crisp looking caustics in dynamic scenes.

A number of extensions to the basic radiosity [8] technique allow specular effects in static scenes [13, 19, 27]. Stochastic approaches to radiosity [3, 21] can be adapted to generate caustics, though like in pathtracing, reducing variance can be expensive. A combination of hierarchical radiosity and particle tracing [9] proved able to render specular effects, like caustics, interactively for simple objects. However, like most particle tracing techniques rendering takes longer for more complex objects.

Using volume data structures to encode lighting information about a scene has been accomplished in the context of static scenes for diffuse [26] and more general reflectance functions [6]. Such volume data structures can illuminate dynamic objects provided they are sufficiently small to not require updates of the volume data structure [10]. However, none of these methods allow a movable specular object to affect the lighting of the scene itself.

Graphics hardware has been used to generate accurate caustics [7, 22]. Recent work [32] has allowed hardware techniques to run interactively. However, hardware techniques limit the type of objects used and multiple passes compound slowdowns from increasing scene complexity. Sampling based hardware approaches [32] only handle focusing from a single specular interaction. Precomputed radiance transfer functions allow graphics hardware to render global illumination effects in real-time [28]. While this technique can render caustics, the results are highly blurred due to use of low-order spherical harmonics and caustics are only cast on predefined objects.

Our technique precomputes all the data required for arbitrarily moving caustics in advance, so a simple table lookup suffices even for complex specular objects. No photon tracing is required between frames, so caustics computations are not dependent on object complexity.

3. Caustics

In this section we describe the caustic behavior and discuss assumptions and simplifications necessary for our technique. Our goal was to develop a method that locally approximates a caustic. We wanted our technique to require little or no recomputation from frame to frame, even when the objects and lights move.

3.1. Caustic Behavior

Caustics are caused by the focusing of light due to reflection or refraction off specular surfaces [23]. Some examples of caustics in daily life include sunlight reflected off

Figure 2. A photo of a real world caustic.

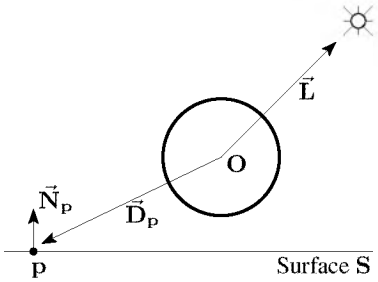

Figure 3. We want to compute the caustic from object O at point p on surface S. This caustic function has 8 dimensions, 3 each from $\vec{D}_p$ and $\vec{L}$, and two from the orientation $\vec{N}_p$ of the receiving surface relative to $\vec{D}_p$.

a watch onto a car ceiling, the cardioid shape at the bottom of a coffee mug (Figure 2), and the focusing of light through a magnifying glass.

While caustics are common, few people know exactly how they should look. For example, one would expect a glass figure to cast a caustic onto a table, but blurred, slightly offset, or even missing details may go unnoticed.

Flat surfaces, like mirrors, reflect light without focusing it. However any concave reflector focuses light into bright lines or points. Technically, only curved surfaces cause caustics, but in this paper, we adopt the common graphics usage and refer to *any* specularly reflected or refracted light as a caustic, as we want to handle both effects.

Consider a transmissive object fixed relative to a lightsource, as in Figure 3. The caustic's intensity at point **p** changes based upon the position of **p** relative to the object and the normal $\vec{N}_p$ of the surface at **p**. For fixed light and object positions, the caustic can be considered a 5D function. Allowing the light (or equivalently the object **O**) to move changes the caustic into a 8D function, by allowing the vector $\vec{L}$ to vary.

Render caustics interactively involves quickly evaluating this 8D caustic intensity function. Unfortunately, analytical descriptions of an arbitrary object's caustic cannot be obtained using current methods, so we fall back to numerically approximating this function.

3.2. Simplifying the Problem

Our simplifications are based on the following observations:

- The direction of incoming light often has greater impact on the visible caustic than distance to the light.
- Lights located relatively far away generate caustics similar to those of lights located infinitely far away.
- Most objects that focus light are relatively far away from the light. The most prevalent exception, mirrors in light fixtures, can usually be treated as part of the lightsource (e.g. Canned Lightsources [11]).
- The most complex caustic behavior usually occurs in regions near the focusing object.

Using these observations, we can make some assumptions to simplify the problem. Combining the first two observations, we assume that the distance to the light source can be ignored. Using directional light sources reduces the dimensionality of the problem by one.

We further assume that some finite volume exists around a reflective or refractive surface in which its caustic contributes significantly to the illumination of other objects. This allows us to sample $\vec{D}_p$ over a finite region. Outside this region, our caustic is based upon samples from the outer region of our sampling volume. Alternately, outside the sampling region caustic contributions could be faded. Note that considering the caustic a local object property limits us to casting caustics onto diffuse surfaces to avoid specularly reflecting the precomputed caustic.

Finally, we assume we can precompute the caustic at **p** for some known surface orientation $\vec{N}_{p_{fixed}}$. We then approximate the caustic for an arbitrary orientation $\vec{N}_p$ by multiplying the precomputed intensity by $\vec{N}_p \cdot \vec{N}_{p_{fixed}}$. We set $\vec{N}_{p_{fixed}} = -\hat{D}_p = -\vec{D}_p / \|\vec{D}_p\|$ at each sample **p**. Conceptually, this method treats all caustic light from **O** as coming from a point light at **O**'s center and computes the caustic intensity at **p** using a cosine falloff based on $\vec{N}_p \cdot -\hat{D}_p$.

Using these assumptions, we can sample a simplified 5D caustic function. These five dimensions are x, y, z, ϕ, and θ, where $\vec{D}_p = (x, y, z)$, and ϕ and θ correspond to the direction of $\hat{L}$.

4. Caustic Sampling

This section outlines the approaches we have examined for sampling and representing the five dimensional caustic

Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)
1550-4085/04 $ 20.00 IEEE

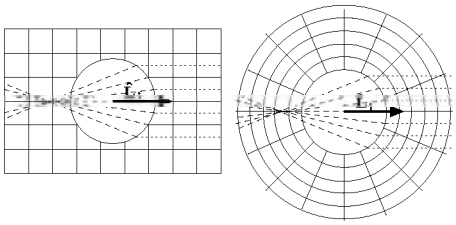

Figure 4. We sampled space either on a uniform grid or a set of concentric shells.

function discussed above. Since our caustics are local properties of an object, sampling must be independently performed on each object which focuses light. We discuss the sampling of the volume over x, y, and z separately from the sampling of incoming light directions ϕ and θ.

4.1. Sampling the Light

For each caustic object, we need to store information about the caustic as the light moves relative to the object. Since we have assumed directional lighting, sampling this lighting is equivalent to sampling directions (ϕ, θ) over a unit sphere.

We found that sampling ϕ and θ in a fixed, uniform or near-uniform, pattern generally works as well as adaptively sampling. Each linear change in ϕ or θ corresponds to varying non-linear changes in the caustic intensity over the volume (x, y, z). Because incoming light often bounces around the object many times, few incoming directions $\hat{\mathbf{L}}$ have "simpler" caustic behavior than others. When we adaptively sampled the sphere, it converged to a relatively uniform sampling.

Currently, we sample ϕ and θ on a subdivided icosahedron. Specifically, we subdivide an icosahedron between 3 and 6 times and project the vertices to the unit sphere. We either sample at the vertices or centers of the subdivided triangles. This provides a nearly uniform sampling over the sphere. We use this method simply because we need not recompute all samples when we subdivide for a denser sampling.

4.2. Sampling Space

Given a light sample $\hat{\mathbf{L}}_\mathbf{i}$, we need to sample the volume around object $\mathbf{O}$. If the object has a bounding volume of radius r, we found in our tests we needed to sample a region with radius $\approx 3r$. However, this varies depending on where focal points of the object lie.

We have sampled this region using two different structures, a uniform grid and a set of concentric shells sampled on a subdivided icosahedron (see Figure 4). After subdividing the volume, we sample the caustic function using the following algorithm. For each light sample $\hat{\mathbf{L}}_\mathbf{i}$, we shoot photons from the directional lightsource towards the object $\mathbf{O}$. Once a photon specularly bounces, it contributes to all the new cells it passes though (the dashed lines in Figure 4).

A photon's contribution to a cell is computed as if it hit a surface at the sample point $\mathbf{p_S}$ with surface normal in the direction of $\mathbf{O}_{center} - \mathbf{p_S}$. At each sample point $\mathbf{p_S}$ we want to store just the caustic intensity (i.e. we do not want to store direct lighting), so we ignore contributions from non-reflected and non-refracted photons. The result is a grid storing approximate irradiance at each cell's center (similar to the Irradiance Volume [10]), with the caveat that only irradiance due to specularly reflected light is stored.

Storing data on a grid has the advantage of easy implementation and fast lookups. However, a rectangular grid structure does not correspond well to caustic data because intensity data changes in a generally radial fashion. This means much space is wasted storing data which changes slowly and not enough is concentrated in regions where the caustic changes quickly.

Storing data on concentric shells allows non-uniform placement of the shells to densely sample the data radially in regions where the caustic varies significantly. Using this allows us to reduce the sampling of one dimension of the volume by up to a factor of 5, either reducing memory usage or allowing a finer sampling in other dimensions. We avoid the difficulty of indexing into the subdivided icosahedron by using a table lookup.

4.3. Data Representation

One of the major problems with sampling a high dimensional function, such as the caustic intensity, is the large storage requirement. Using such data in interactive applications can be difficult if significant portions must remain in memory. We have examined a number of methods to represent this data which reduce the memory overhead. Each approach has its advantages and disadvantages.

Our first implementation stores the complete set of sampled data, both on disk and in memory. Obviously, this requires a machine with lots of memory. For instance, naively storing all sampled data for the ring images (see Figure 11), requires around 1 GB of memory. Our data is stored in colors of three bytes each, one byte for each red, green, and blue channels. The advantage of this technique is easy implementation and fast lookups, leading to faster framerates when data can be completely stored in main memory. Since caustics typically have a large dynamic range, we cannot use these bytes to store the usual values in the range $[0, 1]$. Our examples require values in the $[0, 3]$ range. Using 8-bit values in this way obviously reduces precision, but since

we interpolate to get the irradiance at each point we have not noticed a reduction in quality.

Using a multi-resolution approach helps save memory. We found multi-resolution techniques could reduce memory usage by up to a factor of 10 with equivalent quality results. The tradeoff is that lookups take longer due to the more expensive data traversal routines. This results in moderately reduced framerates. Additionally, multi-resolution approaches may not always reduce storage space.

We also examined using spherical harmonics to compress the sampled data. For each cell in the volume, instead of storing a color for each light sample $\hat{\mathbf{L}}_\mathbf{i}$, we store spherical harmonic coefficients approximating the irradiance for the entire sphere of incoming directions. Alternately, we tried storing one set of spherical harmonic coefficients to represent each of the concentric shells for a given light sample. One main advantage of spherical harmonics is that a large amount of data can be approximated by a few coefficients. The major problem with this approach, however, is that spherical harmonics eliminate most of the high frequency information in a caustic. We believe such sharp features are important to caustic rendering. Increasing the order of the spherical harmonic approximation significantly increases precomputation time as well as the number of coefficients required. As the number of coefficients increases, rendering time slows as well.

5. Caustic Rendering

After sampling our caustic function, we use a raytracer to interactively render the scene. Note that this technique is not limited to raytracers. We simply use a raytracer because it runs interactively on a large shared-memory machine, easily allowing us to access large amounts of memory. Any renderer which can access the necessary data quickly and perform per-pixel operations could use our sampled data to compute caustic intensity.

5.1. Rendering Algorithm

Raytracing the scene proceeds normally until the determination of the color at a diffuse surface. At these surfaces, instead of just looking for direct illumination, we perform lookups into the sampled data to determine if they are illuminated by a caustic. This process can be described algorithmically as follows:

1. Determine the direction $\hat{\mathbf{L}}$ from the center of the object $\mathbf{O}$ to the light. Locate the nearest light sample $\hat{\mathbf{L}}_\mathbf{i}$ (where $\hat{\mathbf{L}} \cdot \hat{\mathbf{L}}_\mathbf{i}$ is maximal). This volume stores the closest approximation to the caustic from the current light position. This step should be done only once per frame, since it is independent of the intersection point $\mathbf{p}$.

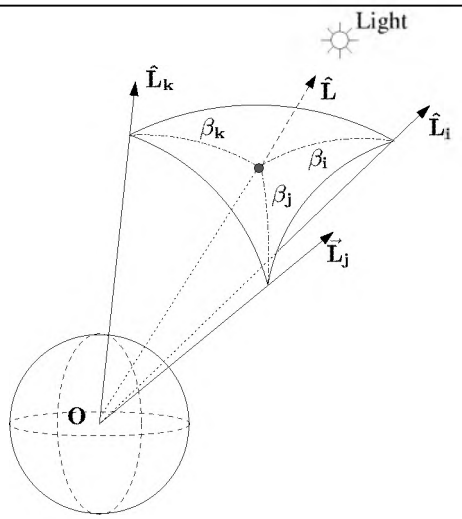

Figure 5. $\hat{\mathbf{L}}$ intersects the spherical triangle formed by $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$.

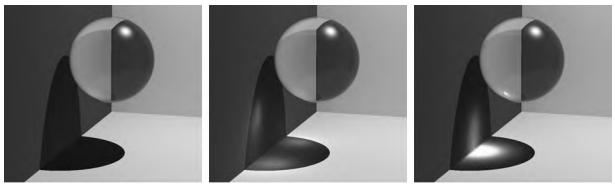

Figure 6. Ghosting happens when the caustic changes significantly between neighboring light samples $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$. Images (left) without caustics, (center) with ghost caustics, and (right) a correct caustic.

2. At each intersection point $\mathbf{p}$, find $\mathbf{p}$'s location in the volume sampled around $\mathbf{O}$ and look up the caustic contribution. Add this result to the direct lighting computed by the renderer.

5.2. Issues Rendering Caustic Data

Unfortunately, using a single light sample $\hat{\mathbf{L}}_\mathbf{i}$ to render the caustic causes temporal coherence issues as objects move. This is due to differences in the caustic from one light sample to the next.The popping can be reduced by combining the caustic from multiple light samples $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$ (where $\hat{\mathbf{L}} \cdot \hat{\mathbf{L}}_\mathbf{i} \geq \hat{\mathbf{L}} \cdot \hat{\mathbf{L}}_\mathbf{j} \geq \hat{\mathbf{L}} \cdot \hat{\mathbf{L}}_\mathbf{k} \geq \hat{\mathbf{L}} \cdot \hat{\mathbf{L}}_\mathbf{m}, \forall m \notin \{i, j, k\}$). $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$ form the three vertices of a spherical triangle on the unit sphere which includes $\hat{\mathbf{L}}$ (see Figure 5).

Combining three light samples using barycentric coordinates [4] eliminates popping between caustic samples but

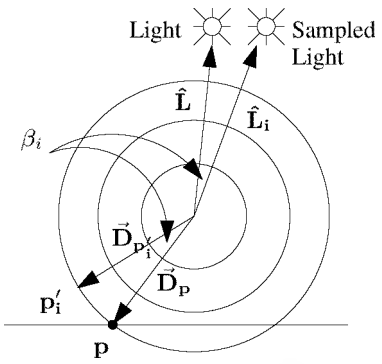

Figure 7. Find the cell to use in the average by rotating $\vec{\mathbf{D}}_\mathbf{p}$ around the axis $\vec{\mathbf{R}}_\mathbf{i}$ (which points into the page at $\mathbf{O}_{\mathbf{center}}$) by angle β_i.

introduces a new problem—ghosting (see Figure 6). Ghosting happens because object $\mathbf{O}$'s caustic can differ significantly between neighboring light samples, so blending data from $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$ results in three separate faint caustics. Unfortunately, the best way to eliminate ghosting is to sample the caustic for more light directions. This significantly increases memory consumption.

Below, we describe a technique which we found helps reduce ghosting for relatively smooth objects. This algorithm replaces step 2 from the rendering algorithm described above:

A. Compute the vector $\vec{\mathbf{D}}_\mathbf{p}$ from $\mathbf{O}_{center}$ to $\mathbf{p}$.

B. Find the barycentric coordinates of $\hat{\mathbf{L}}$ in the spherical triangle formed by $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$. This gives the relative contributions from each light sample (Figure 5).

C. Compute the angles β_i, β_j, and β_k between $\hat{\mathbf{L}}$ and the three nearest sampled light directions $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, and $\hat{\mathbf{L}}_\mathbf{k}$.

D. Calculate rotation axes $\vec{\mathbf{R}}_\mathbf{i}$, $\vec{\mathbf{R}}_\mathbf{j}$, and $\vec{\mathbf{R}}_\mathbf{k}$ by taking the cross product between $\vec{\mathbf{L}}$ and $\vec{\mathbf{L}}_\mathbf{i}$, $\vec{\mathbf{L}}_\mathbf{j}$, and $\vec{\mathbf{L}}_\mathbf{k}$, respectively.

E. Rotate vector $\vec{\mathbf{D}}_\mathbf{p}$ around the axes $\vec{\mathbf{R}}_\mathbf{i}$ by angle β_i to find a new vector $\vec{\mathbf{D}}_{\mathbf{p}'_\mathbf{i}}$. Similarly find $\vec{\mathbf{D}}_{\mathbf{p}'_\mathbf{j}}$ and $\vec{\mathbf{D}}_{\mathbf{p}'_\mathbf{k}}$ by rotating around $\vec{\mathbf{R}}_\mathbf{j}$ and $\vec{\mathbf{R}}_\mathbf{k}$ by angles β_j and β_k (Figure 7).

F. Find points $\mathbf{p}'_\mathbf{i}$, $\mathbf{p}'_\mathbf{j}$, and $\mathbf{p}'_\mathbf{k}$. Where $\mathbf{p}'_\mathbf{i} = \mathbf{O}_{center} + \vec{\mathbf{D}}_{\mathbf{p}'_\mathbf{i}}$.

G. Perform caustic lookups as if $\mathbf{p}'_\mathbf{i}$, $\mathbf{p}'_\mathbf{j}$, and $\mathbf{p}'_\mathbf{k}$ were the intersection points (instead of $\mathbf{p}$). Weight the contributions from these points based on the barycentric coordinates computed in step B.

The process performs an interpolation between samples. Unfortunately, such an interpolation is not generally valid, as it assumes the caustic changes linearly in space for a linear change in light direction. We found for relatively smooth objects, like the sphere and bunny, such "interpolation" generally allows us to use fewer light samples. For objects such as the cube and prism which have sharp angles, we found that this approach does not significantly reduce ghosting.

When using spherical harmonic coefficients, which essentially filter over the sampled light directions, we can avoid this rendering process and instead render using the approach outlined in Sloan et al. [28]. The downside to this approach is blurred caustics.

	Sphere	Cube	Prism	Ring	Building	Bunny
Grid Caustics (fps)	15.2	12.1	12.7	9.3	1.94	2.16
Shell Caustics (fps)	17.3	12.6	13.2	9.5	2.01	2.30
Multi-Resolution Caustics (fps)	15.0	10.8	11.0	8.8	1.90	2.25
5th Ord. Sph. Harm. Caustics (fps)	8.1	6.1	6.5	5.2	1.48	1.80
No Caustics (fps)	26.9	20.2	20.3	12.9	2.29	2.55
Shoot Photons (per sample) (sec) on 1 CPU	1.7	2.4	2.0	1.3	4.5	25.0

Table 1. Framerates are for thirty 400 MHz R12000 processors rendering a 360^2 window. Times for shooting photons are for a single 400 MHz R12000 processor.

6. Results

We implemented our algorithm on an interactive parallel raytracer running on an SGI Origin 3800 with thirty-two 400 MHz R12000 processors. This is a shared memory machine which easily holds our entire scene and caustic datasets in main memory. However, our approach is not limited to such applications. Any renderer which has per-pixel lighting control could implement our technique given enough memory. Existing systems (e.g. [9, 24, 29, 31, 30]) could easily incorporate our method to avoid the cost of reshooting photons each frame.

Table 1 contains timings for the images generated for Figures 1, 8, and 9. We incur a 10–45% speed penalty for displaying caustics, depending on the relative costs of the caustic lookups to the raytracing costs of the scene. The cost of our photon shooting preprocess ranges from 1.3 to 25 seconds per light sample using a single CPU. Shooting photons for a photon map takes a similar amount of time, though additional overhead is needed to create the required kd-tree. Framerates are for a 360 x 360 window running on thirty processors.

Figure 8 compares a photon map with our technique using both the grid and concentric shell storage techniques. The comparisons are between grids and shells using roughly the same memory. We show data compressed using 5th and 15th order spherical harmonics. The advantages of the

Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04)
1550-4085/04 $ 20.00 IEEE

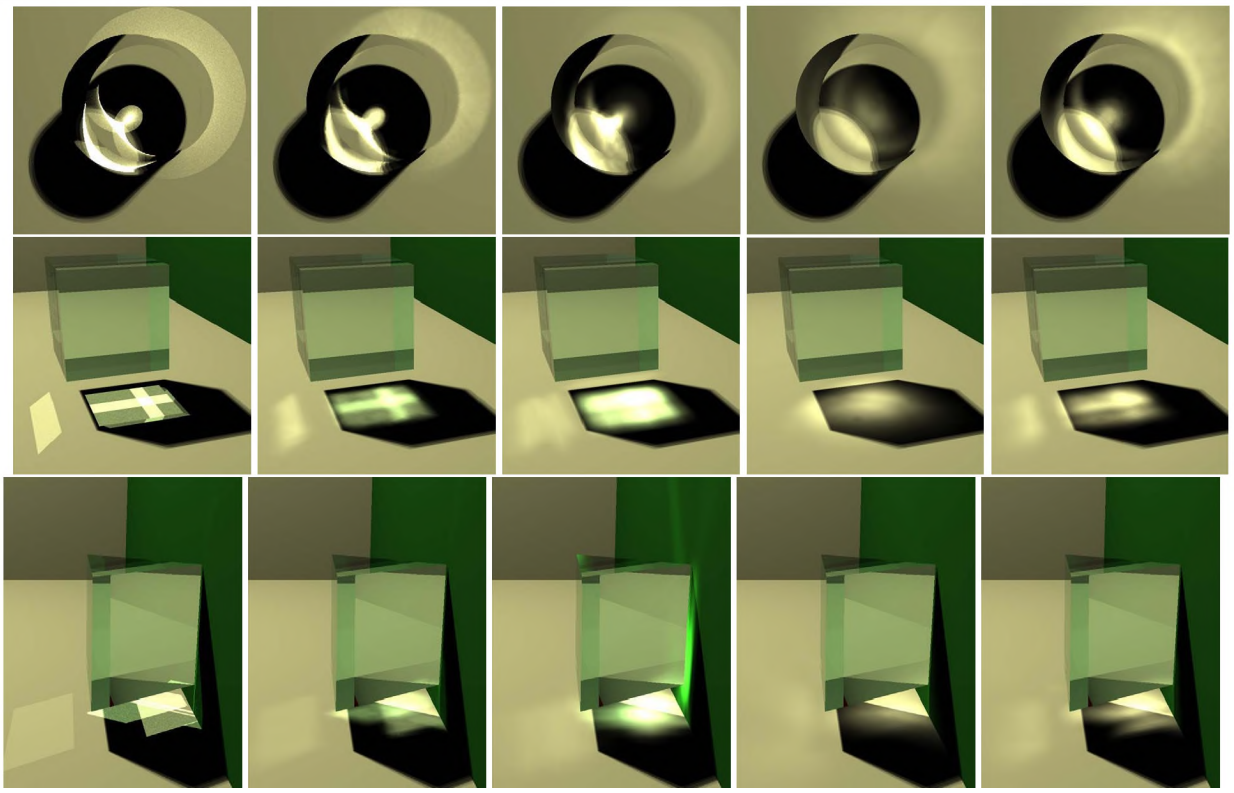

Figure 8. From left to right: Images generated with a photon map, our concentric shell approach, our grid technique, and 5th and 15th order spherical harmonic representations.

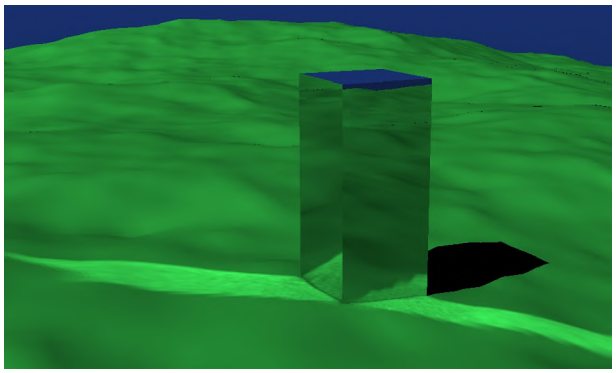

Figure 9. Caustics can also be dynamically cast on complex terrain.

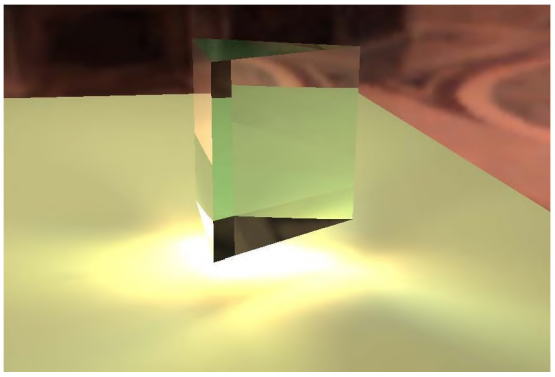

Figure 10. The caustic of a prism in St. Peter's cathedral using 5th order spherical harmonics. Note there are no shadows in this image.

spherical harmonic representation are the ability to use area lights (see Figure 10) and its high temporal coherence and low memory consumption. For comparison, these 5th order representations use nearly 10 MB memory, about as much as the uncompressed data used for Figure 11a. Since the number of coefficients increases quadratically with order, computation costs quickly become the bottleneck. All our scenes run at less than 1 frame per second with 15th order spherical harmonics.

Figure 11 illustrates the effect of sampling density on memory consumption and caustic quality. For relatively smooth object, like the bunny (see Figure 1), we used 162

Figure 11. Sharper caustics come at the expense of denser sampling. The images shown use 5.7, 22.5, 90.1, 360, and 1440 kB of memory per light sample using the concentric shell representation. Similar quality with the multiresolution approach requires 4.8, 12.6, 29.5, 70.4, and 179 kB of memory per light sample.

light samples. For objects where our rotational alignment technique from Section 5.2 does not work well (like the cube and prism), we needed up to 2500 light samples. Note that number of light samples does not affect framerate, assuming the data can all fit into memory.

Obviously, with symmetric objects one need not sample the entire sphere of incoming light directions. For a sphere, a single sample suffices. For the metallic ring, we found between 50 and 100 light samples are sufficient for good temporal coherence. Many common objects have symmetrical properties which could be used to simplify sampling.

7. Conclusions

We have presented a novel technique for rendering approximate caustics interactively by localizing the problem to the vicinity of the focusing object. This approach avoids the recurring cost of photon shooting which existing methods require to generate dynamic caustics. Because particle tracing in not necessary between frames, this technique could be applied to other interactive systems that cannot traditionally perform such computations (e.g. hardware based renderers). Additionally, the rendering costs of our method are independent of object complex. We examined a number of ways of sampling the data and representing the samples in memory. Since our method generates caustics using table lookups, memory becomes the bottleneck.

We have found that storing a highly sampled caustic function in memory produces the best looking results. Unfortunately, the memory requirements make the technique difficult to use unless object symmetries or other simplifying conditions exist. Multi-resolution approaches can significantly reduce memory overhead by storing densely sampled data only where necessary. In exchange lookups are more costly.

Storing data using spherical harmonics generally blurs caustics extensively. We believe that the results look unconvincing, though memory requirements are modest enough to allow implementation on current graphics hardware. Higher order approximations improve results at the expense of additional coefficients. We plan on examining other bases, such as spherical wavelets, to see if they result in sharper caustics with similar memory savings.

Scenes that lend themselves well to our technique include outdoors scenes where the sun effectively acts as a constant directional lightsource. Such a scene requires a single light sample. Leveraging object symmetries also can reduce some of the memory burden. Many common objects have such symmetries, so our sampling techniques may be feasible for such objects.

Our work has a number of limitations, including:

- Expensive memory requirements for general environments when the entire sampled dataset must be available.
- Poor realignment of neighboring light samples causes ghosting when ϕ and θ are not sampled densely enough.
- Area light sources are not handled without spherical harmonics. Since the shape of a light can significantly affect the caustics, this problem needs to be addressed.
- Our assumptions rule out using this method for scenes with reflective or refractive objects near the lights.

We believe that the alignment of light samples presents a serious problem, particularly for objects with large planar surfaces. We plan on examining ways of representing the entire 5D dataset instead of simply considering the function as a 2D array of 3D volumes. Such a representation may allow us to perform a true interpolation between light samples. Such interpolation would eliminate the need for a dense sampling of $\phi - \theta$ space.

Current graphics hardware has extensive pixel shader hardware which could apply our sampled data in interactive OpenGL or DirectX applications. We plan on examining the details involved with such an approach.

We believe that global illumination gives important information to users of interactive systems and cannot be ignored. Our results indicate that viable techniques exist for including specular effects in addition to diffuse global illumination in these applications.

References

[1] J. Arvo. Backward ray tracing. *Developments in Ray Tracing*, pages 259–263, 1986. ACM Siggraph '86 Course Notes.

[2] K. Bala, J. Dorsey, and S. Teller. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Transactions on Graphics*, 18(3):100–130, August 1999.

[3] P. Bekaert. *Hierarchical and stochastic algorithms for radiosity*. PhD thesis, Department of Computer Science, Kaltholieke Universiteit Leuven, 1999.

[4] B. Cabral, M. Olano, and P. Nemec. Reflection space image based rendering. In *Proceedings of SIGGRAPH*, pages 165–170, 1999.

[5] M. Chen and J. Arvo. Theory and application of specular path perturbation. *ACM Transactions on Graphics*, 19(4):246–278, October 2000.

[6] K. Chiu, K. Zimmerman, and P. Shirley. The light volume: an aid to rendering complex environments. In *Eurographics Rendering Workshop*, pages 1–10, 1995.

[7] P. J. Diefenbach and N. I. Badler. Multi-pass pipeline rendering: Realism for dynamic environments. In *ACM Symposium on Interactive 3D Graphics*, pages 59–70, 1997.

[8] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. In *Proceedings of SIGGRAPH*, pages 213–222, 1984.

[9] X. Granier, G. Drettakis, and B. Walter. Fast global illumination including specular effects. In *Eurographics Rendering Workshop*, pages 47–58, 2000.

[10] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, March-April 1998.

[11] W. Heidrich, J. Kautz, P. Slusallek, and H.-P. Seidel. Canned lightsources. In *Eurographics Rendering Workshop*, pages 293–300, 1998.

[12] H. Hu, A. Gooch, W. Thompson, B. Smits, J. Rieser, and P. Shirley. Visual cues for imminent object contact in realistic virtual environments. In *Proceedings of Visualization*, pages 127–136, 2000.

[13] D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of SIGGRAPH*, pages 133–142, 1986.

[14] H. W. Jensen. Importance driven path tracing using the photon map. In *Eurographics Rendering Workshop*, pages 326–335, 1995.

[15] H. W. Jensen. Global illumination using photon maps. In *Eurographics Rendering Workshop*, pages 21–30, 1996.

[16] J. T. Kajiya. The rendering equation. In *Proceedings of SIGGRAPH*, pages 143–150, 1986.

[17] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff. Illusory motion from shadows. *Nature*, 279(6560):31, 1996.

[18] G. W. Larson and M. Simmons. The holodeck ray cache: An interactive rendering system for global illumination in non-diffuse environments. *ACM Transactions on Graphics*, 18(4):361–368, October 1999.

[19] T. J. V. Malley. A shading method for computer generated images. Master's thesis, Computer Science Department, University of Utah, June 1988.

[20] D. P. Mitchell and P. Hanrahan. Illumination from curved reflectors. In *Proceedings of SIGGRAPH*, pages 283–291, 1992.

[21] L. Neumann, M. Feda, M. Kopp, and W. Purgathofer. A new stochastic radiosity method for highly complex scenes. In *Eurographics Rendering Workshop*, pages 201–213, 1994.

[22] T. Nishita and E. Nakamae. Method of displaying optical effects within water using accumulation buffer. In *Proceedings of SIGGRAPH*, pages 373–381, 1994.

[23] J. F. Nye. *Natural Focusing and Fine Structure of Light*. Institute of Physics Publishing, Bristol, 1999.

[24] S. Parker, W. Martin, P.-P. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. In *ACM Symposium on Interactive 3D Graphics*, pages 119–126, 1999.

[25] T. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIOGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, pages 41–50, 2003.

[26] H. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings of Graphics Interface*, pages 227–236, 1993.

[27] H. Rushmeier and K. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Transactions on Graphics*, 9(1):1–27, January 1990.

[28] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions of Graphics*, 21(4):527–536, July 2002.

[29] P. Tole, F. Pellacini, B. Walter, and D. P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions of Graphics*, 21(4):537–546, July 2002.

[30] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination using fast ray tracing. In *Eurographics Rendering Workshop*, pages 15–24, 2002.

[31] B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. In *Eurographics Rendering Workshop*, pages 19–30, Granada, Spain, June 1999. Springer Wein / Eurographics.

[32] M. Wand and W. Straßer. Real-time caustics. *Computer Graphics Forum*, 22(3):611–620, 2003.