Interactive Raytraced Caustics

Chris Wyman Charles Hansen Peter Shirley

University of Utah, School of Computing
Technical Report UUCS-03-009

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

April 23, 2003

Abstract

In computer graphics, bright patterns of light focused 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.

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, too much effort is spent computing illumination that has only a minor impact on the image and a negligible perceptual impact. In fact, while global illumination provides humans perceptual cues as to relative object locations, accuracy is not always important[1, 2].

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, computing the contribution of sunlight reflected off a wooden pencil onto the wall across the room is an academic exercise, as the pencil's contribution on any but the nearest objects is small. While some researchers[3] have looked into simplifying the environment to reduce unnecessary computations, significant questions remain as to how much simplification will compromise the perceived quality of the global illumination.

In this paper, we examine the focusing of light caused by reflective and refractive surfaces. This focusing, known in computer graphics as a "caustic," potentially affects the entire environment. However, in most cases caustics are seen in a relatively localized space around the objects causing them. For example, one might see a caustic from a glass figurine on a table or the caustic from a mirror on an adjacent wall.

Our technique samples the caustic near the focusing object. This allows us to reduce caustic rendering from a global problem to a localized property which can be computed with a simple lookup. We can perform this lookup at interactive framerates even when objects or 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. In Section 2, we outline the previous work in computing and speeding up global illumination. Section 3 discusses the behavior of

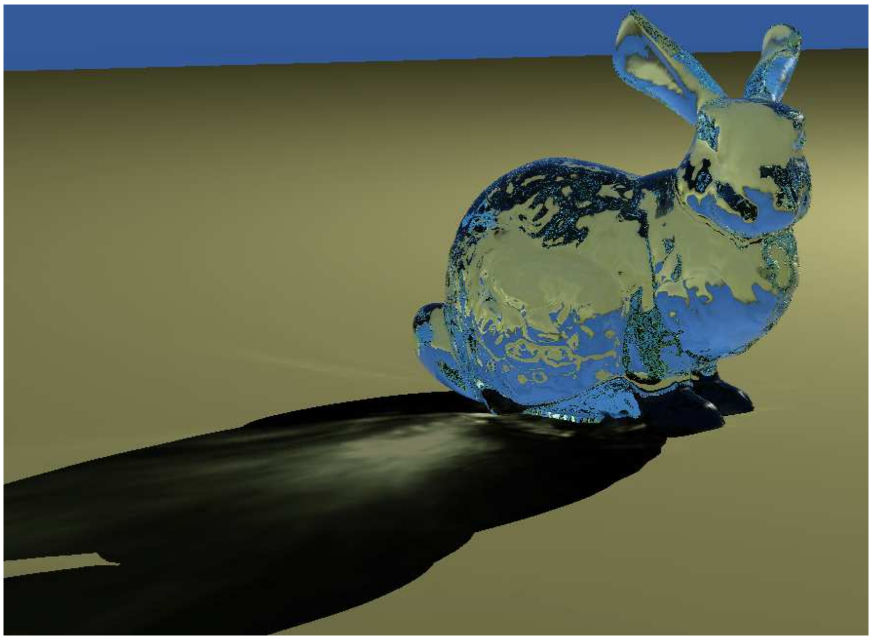

Figure 1: *Our technique generates this caustic at 2.3 fps on 30 processors while moving the bunny or light.*

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.

Extremely accurate caustics have been generated semi-analytically for smooth surfaces[4], but that method is too slow for interactivity. Also, researchers have investigated accurate interpolation between specular rays[5, 6], but these techniques have not yet yielded fast sampling-based methods for accurate caustic generation.

Pathtracing[7] generates beautiful global illumination renderings, but accuracy comes at an extreme computational cost. Numerous researchers have looked into speeding up pathtracing and raytracing[8, 9, 10, 11]. 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[12]. Many researchers have since used this technique, and it has been extended to include non-diffuse surfaces (e.g., photon mapping[13, 14]). 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. Combinations of photon shooting and pathtracing have been examined[15]. 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[16] technique allow specular effects in static scenes[17, 18, 19]. Stochastic approaches to radiosity[20, 21] can be adapted to generate caustics, though like pathtracing reducing variance can be expensive. A combination of hierarchical radiosity and particle tracing[22] 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[3] and more general reflectance functions[23]. It has been shown that such data structures can be used to illuminate dynamic objects provided they are sufficiently small to not require updates of the volume data structure[24]. 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[25, 26]. However, such techniques are far from interactive and limit the use of curved reflectors and refractors. Precomputed radiance transfer functions allow graphics hardware to render global illumination effects in real-time[27]. While this technique can render caustics, results are highly blurred due to the use of low-order spherical harmonics.

Our technique precomputes all the data required for arbitrarily moving caustics in advance,

Figure 2: *A photograph of a real world caustic.*

so a simple table lookup suffices even for complex specular objects. No photon tracing is required between frames, so caustics computations are not dependant on object complexity.

3 Caustics

In this section we describe the behavior of caustics and discuss the 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[28]. Some examples of caustics in daily life include sunlight reflected off 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.

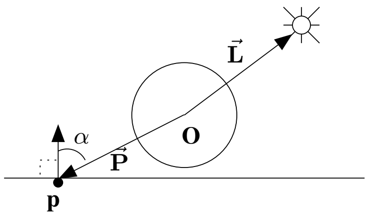

Figure 3: *We want to compute the caustic from object* $\mathbf{O}$ *at* $\mathbf{p}$. *This caustic function has 8 dimensions, 3 each from* $\vec{\mathbf{P}}$ *and* $\vec{\mathbf{L}}$*, and two from the orientation* α *of the receiving surface relative to* $\vec{\mathbf{P}}$.

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. The caustic's intensity at point **p** changes based upon the position of **p** relative to the object and the orientation α 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 $\vec{\mathbf{L}}$ to vary (see Figure 3).

To render caustics interactively, we must be able to quickly evaluate this 8D caustic intensity function. Unfortunately no current methods allow the extraction of a caustic's analytic description from an arbitrary object, so we fall back to numerically approximating the function.

3.2 Simplifying the Problem

Our simplifications are based on the following observations:

- The direction to the light often has a greater impact on the visible caustic than the 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[29]).

- The area nearby a object generally contains its most complex caustic behavior.

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. This allows us to reduce the dimensionality of the problem by one by assuming that all lights are directional.

We further assume that some finite volume exists around the reflective or refractive surface in which its caustic contributes significantly to the illumination of other objects. This allows us to sample $\vec{\mathbf{P}}$ 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 for some known orientation α_{fixed}. We can then compute the caustic intensity at **p** using the cosine of the difference between α_p and α_{fixed}. We set $\alpha_{fixed} = -\hat{\mathbf{P}} = -\vec{\mathbf{P}}/\|\vec{\mathbf{P}}\|$ at each sample.

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

4 Caustic Sampling

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

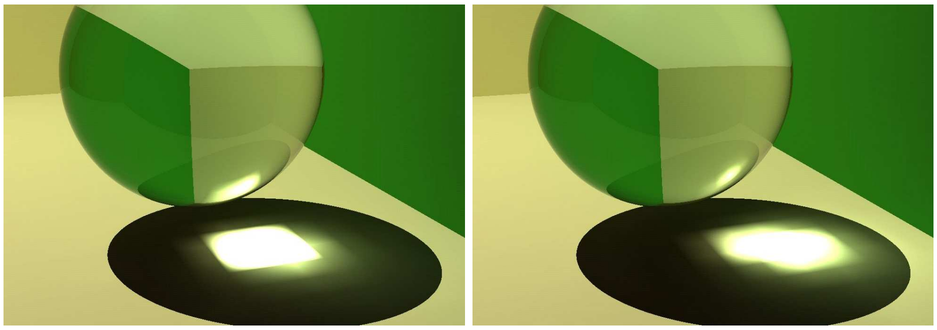

Figure 4: *A linear change in ϕ and θ does not correspond to a linear change in the caustic.*

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 (see Figure 4) in the caustic intensity over the volume (x, y, z). Because incoming light often bounces around the object many times, few incoming directions $\mathbf{\hat{L}}$ have a "simpler" caustic behavior than others. Thus, adaptive sampling of the sphere tends to converge to a relatively uniform sampling.

Currently, we sample ϕ and θ on a geodesic. Specifically, we subdivide an icosahedron between 3 and 6 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 $\mathbf{\hat{L}_i}$, 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 subdivided as a geodesic (see Figure 5). After subdividing the volume, we sample the caustic function using the following algorithm. For each light sample $\mathbf{\hat{L}_i}$, 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 5).

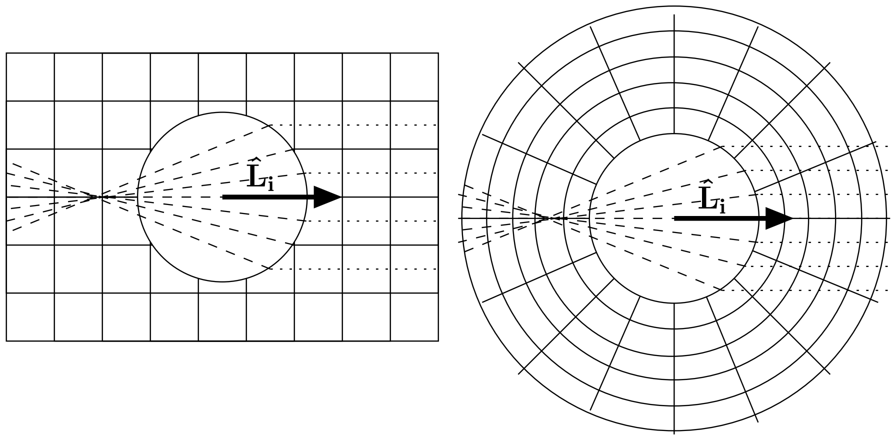

Figure 5: *We sampled space either on a uniform grid or a set of concentric shells.*

A photon's contribution to a cell is computed as if it hit a surface at the sample point S with surface normal in the direction of $\mathbf{O}_{center} - S$. After shooting photons, the values in each cell must be normalized by how many photons would have hit the cell without object **O** present. The idea is that each grid cell stores an approximation of the irradiance at the cell's center (similar to the Irradiance Volume[24]), with the caveat that we only store irradiance due to specularly reflected light.

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 a geodesic 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.

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.

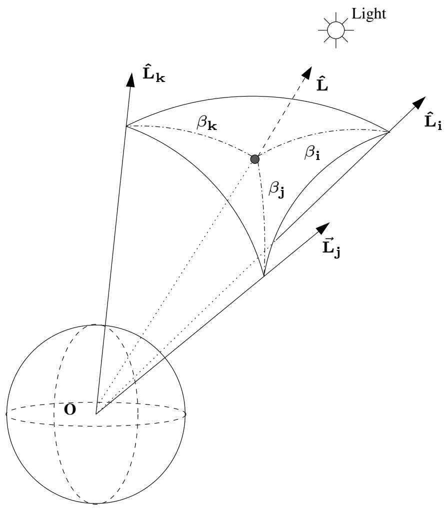

Figure 6: $\hat{\mathbf{L}}$ *intersects the spherical triangle formed by* $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, *and* $\hat{\mathbf{L}}_\mathbf{k}$.

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 **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 **p**.

2. At each intersection point **p**, find **p**'s location in the volume sampled around **O** and look up the caustic contribution. Add this result to the direct lighting computed by the raytracer.

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 (see Figure 4). 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 $\hat{\mathbf{L}}\cdot\hat{\mathbf{L}}_\mathbf{i} \geq \hat{\mathbf{L}}\cdot\hat{\mathbf{L}}_\mathbf{j} \geq \hat{\mathbf{L}}\cdot\hat{\mathbf{L}}_\mathbf{k} \geq \hat{\mathbf{L}}\cdot\hat{\mathbf{L}}_\mathbf{m}, \forall m \notin \{i, j, k\}$).

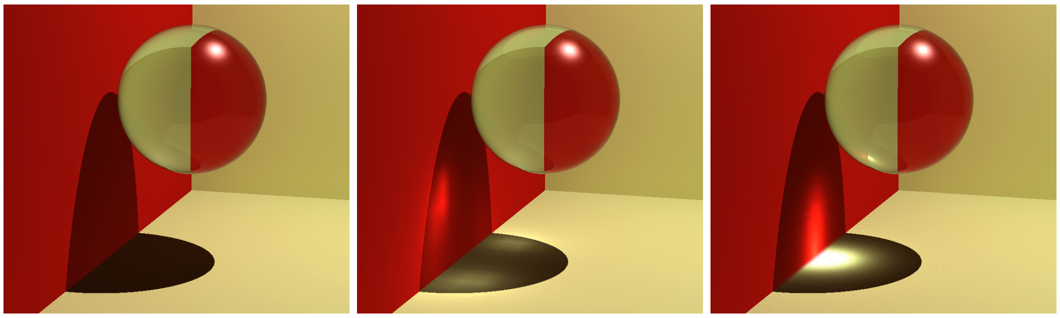

Figure 7: *Ghosting happens when the caustic changes significantly between neighboring light samples* $\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, *and* $\hat{\mathbf{L}}_\mathbf{k}$. *Images (left) without caustics, (center) with ghost caustics, and (right) a correct caustic.*

$\hat{\mathbf{L}}_\mathbf{i}$, $\hat{\mathbf{L}}_\mathbf{j}$, 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 6).

Using three light samples eliminates popping between caustic samples but introduces a new problem—ghosting (see Figure 7). Ghosting happens because object **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 $\vec{\mathbf{P}}$ 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 6).

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 $\vec{\mathbf{P}}$ around the axes $\vec{\mathbf{R}}_\mathbf{i}$ by angle β_i to find a new vector $\vec{\mathbf{P}}'_\mathbf{i}$. Similarly find $\vec{\mathbf{P}}'_\mathbf{j}$ and $\vec{\mathbf{P}}'_\mathbf{k}$ by rotating around $\vec{\mathbf{R}}_\mathbf{j}$ and $\vec{\mathbf{R}}_\mathbf{k}$ by angles β_j and β_k (Figure 8).

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

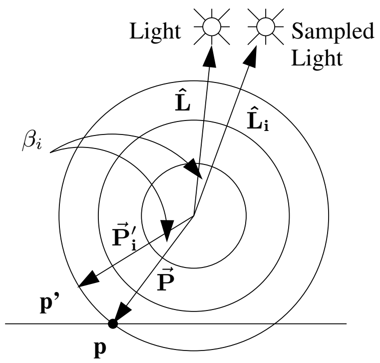

Figure 8: *Find the cell to use in the weighted average by rotating* $\vec{\mathbf{P}}$ *around the axis* $\vec{\mathbf{R}}_\mathbf{i}$ *(which points into the page at* $\mathbf{O_{center}}$*) by angle* β_i*.*

G. Perform caustic lookups as if $\mathbf{p'_i}$, $\mathbf{p'_j}$, and $\mathbf{p'_k}$ 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 reduce the ghosting.

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. [22, 9, 10, 11, 15]) could easily incorporate our method to avoid the cost of reshooting the photons causing caustics each frame.

Table 1 contains timings for the images generated for Figures 1, 9, and 10. We incurr 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

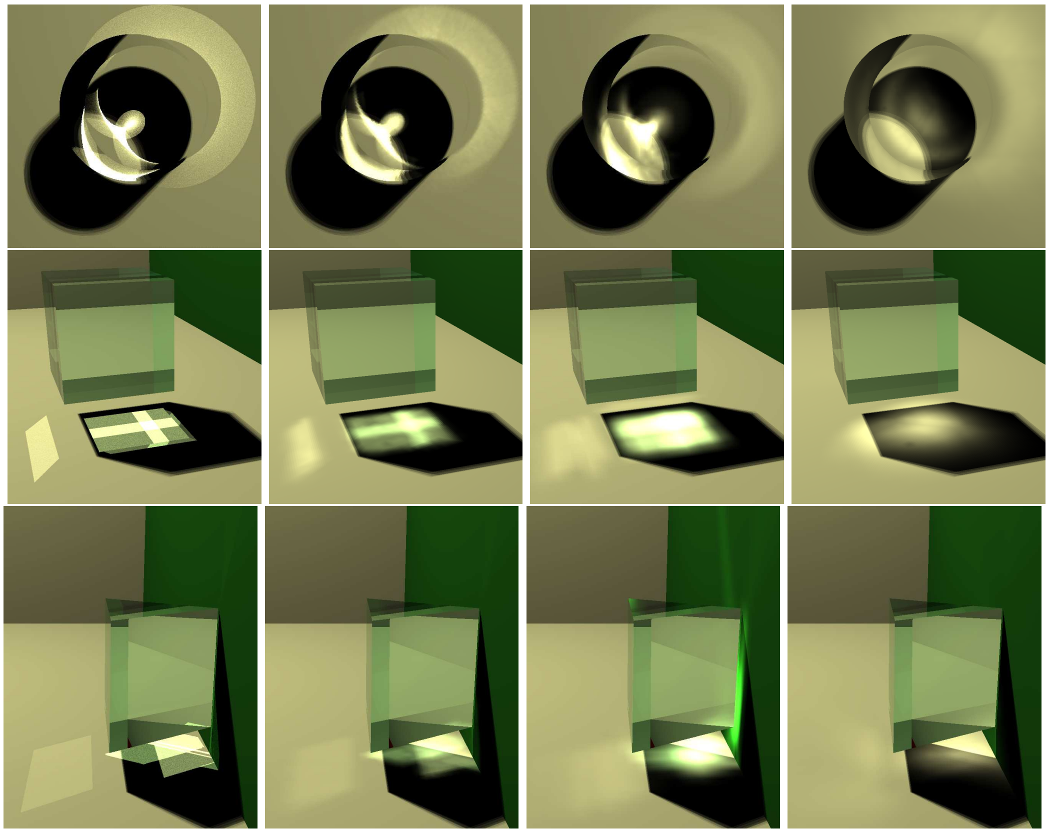

Figure 9: *From left to right: Images generated with a photon map, our concentric shell approach, our grid technique, and a 5th order spherical harmonic representation.*

ranges from 1.3 to 25 seconds per light sample. 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 9 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 a 5th order spherical harmonic, which runs at roughly the same speed as indexing into sampled data. The advantage of the spherical harmonic representation is its high temporal coherence and low memory consumption. For comparison, these 5th order representations require as much memory as the data used for Figure 11a. Since the number of coefficients increases quadratically with order, computation costs quickly become the bottleneck.

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 light

Object	Grid Caustics (fps)	Shell Caustics (fps)	MultiRes Caustics (fps)	No Caustics (fps)	Shoot Photons (per sample) (sec)
Sphere	15.2	17.3	15.0	26.9	1.7
Cube	12.1	12.6	10.8	20.2	2.4
Prism	12.7	13.2	11.0	20.3	2.0
Ring	9.3	9.5	8.8	12.9	1.3
Building	1.94	2.01	1.90	2.29	4.5
Bunny	2.16	2.30	2.25	2.55	25.0

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

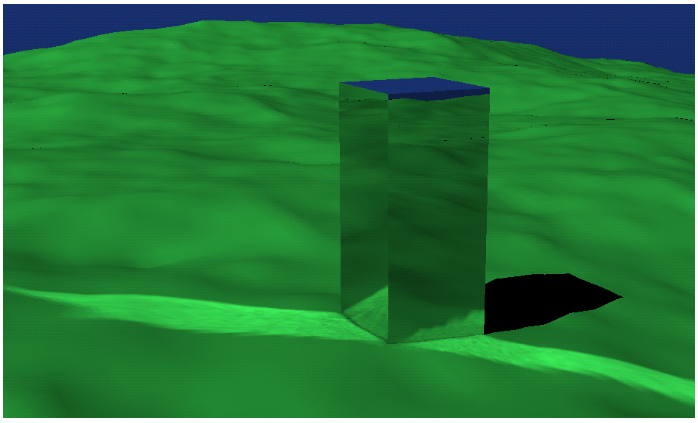

Figure 10: *This "building" can dynamically cast caustics on surrounding terrain based on the sun's position.*

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 the sampling space.

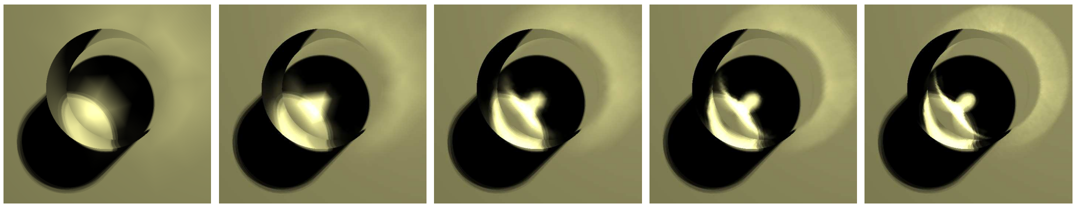

Figure 11: *Sharper caustics come at the expense of denser sampling. The images shown require 5.7, 22.5, 90.1, 360, and 1440 kilobytes of memory per light sample. The data is stored using the concentric shell representation. Using a multiresolution approach, similar results require 4.8, 12.6, 29.5, 70.4, and 179 kilobytes of memory per light sample.*

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 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 independant 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. Higher order approximations will 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. 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.

Acknowledgments

The authors would like to thank the reviewers and countless other people who provided suggestions on the work and gave valuable feedback on the text. This material is based upon work supported by the National Science Foundation under Grants: 9977218 and 9978099.

References

[1] 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*, pp. 127–136, 2000.

[2] D. Kersten, D. C. Knill, P. Mamassian, and I. Bulthoff, "Illusory motion from shadows," *Nature*, vol. 279, no. 6560, p. 31, 1996.

[3] H. Rushmeier, C. Patterson, and A. Veerasamy, "Geometric simplification for indirect illumination calculations," in *Proceedings of Graphics Interface*, pp. 227–236, 1993.

[4] D. P. Mitchell and P. Hanrahan, "Illumination from curved reflectors," in *Proceedings of SIGGRAPH*, (Chicago, Illinois), pp. 283–291, 1992.

[5] K. Bala, J. Dorsey, and S. Teller, "Radiance interpolants for accelerated bounded-error ray tracing," *ACM Transactions on Graphics*, vol. 18, pp. 100–130, August 1999.

[6] M. Chen and J. Arvo, "Theory and application of specular path perturbation," *ACM Transactions on Graphics*, vol. 19, pp. 246–278, October 2000.

[7] J. T. Kajiya, "The rendering equation," in *Proceedings of SIGGRAPH*, pp. 143–150, 1986.

[8] 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*, vol. 18, pp. 361–368, October 1999.

[9] S. Parker, W. Martin, P.-P. J. Sloan, P. S. Shirley, B. Smits, and C. Hansen, "Interactive ray tracing," in *ACM Symposium on Interactive 3D Graphics*, pp. 119–126, 1999.

[10] P. Tole, F. Pellacini, B. Walter, and D. Greenburg, "Interactive global illumination in dynamic scenes," in *Proceedings of SIGGRAPH*, pp. 537–546, 2002.

[11] B. Walter, G. Drettakis, and S. Parker, "Interactive rendering using the render cache," in *Eurographics Rendering Workshop 1999*, (Granada, Spain), pp. 19–30, Springer Wein / Eurographics, June 1999.

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

[13] H. W. Jensen, "Importance driven path tracing using the photon map," in *Eurographics Rendering Workshop*, pp. 326–335, 1995.

[14] H. W. Jensen, "Global illumination using photon maps," in *Eurographics Rendering Workshop*, pp. 21–30, 1996.

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

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

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

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

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

[20] P. Bekaert, *Hierarchical and Stochastic Algorithms for Radiosity*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, 1999.

[21] L. Neumann, M. Feda, M. Kopp, and W. Purgathofer, "A new stochastic radiosity method for highly complex scenes," in *Eurographics Rendering Workshop*, pp. 283–291, 1994.

[22] X. Granier, G. Drettakis, and B. Walter, "Fast global illumination including specular effects," in *Eurographics Rendering Workshop*, pp. 47–58, 2000.

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

[24] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg, "The irradiance volume," *IEEE Computer Graphics & Applications*, vol. 18, pp. 32–43, March-April 1998.

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

[26] T. Nishita and E. Nakamae, "Method of displaying optical effects within water using accumulation buffer," in *Proceedings of SIGGRAPH*, pp. 373–381, 1994.

[27] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," in *Proceedings of SIGGRAPH*, pp. 527–536, 2002.

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

[29] W. Heidrich, J. Kautz, P. Slusallek, and H.-P. Seidel, "Canned lightsources," in *Eurographics Rendering Workshop*, pp. 293–300, 1998.