This paper appeared in *Proceedings of Graphics Interface '96*, pages 116-121, 1996

Rendering Caustics on Non-Lambertian Surfaces

Henrik Wann Jensen

Department of Graphical Communication
The Technical University of Denmark
hwj@gk.dtu.dk, http://www.gk.dtu.dk/~hwj

Abstract

This paper presents a new technique for rendering caustics on non-Lambertian surfaces. The method is based on an extension of the photon map which removes previous restrictions limiting the usage to Lambertian surfaces. We add information about the incoming direction to the photons and this allows us to combine the photon map with arbitrary reflectance functions. Furthermore we introduce balancing of the photon map which not only reduces the memory requirements but also significantly reduces the rendering time. We have used the method to render caustics on surfaces with reflectance functions varying from Lambertian to glossy specular.

Keywords: Caustics, Photon Map, Ray Tracing, Rendering.

1 Introduction

Caustics provides some of the most spectacular patterns of light in nature. Caustics are formed when light reflected from or transmitted through a specular surfaces strikes a diffuse surface. An example is the caustic formed as light shines through a glass of wine onto a table.

In traditional ray tracing [22] diffuse surfaces are only illuminated by the light sources. Caustics which are indirect illumination on the diffuse surfaces are not rendered at all. Even the stochastic ray tracing methods [5, 8] cannot

render caustics properly. In order to integrate the computation of caustics into ray tracing it is necessary to compute illumination from light transmitted via specular surfaces. This computation is in most situations very complex and it has been solved only for a simple class of specular objects (ie. polygons [20]). Mitchell et al. [12] has presented a very comprehensive technique and their method is capable of handling caustics from implicit surfaces. The method is unfortunately very complex and also very time consuming.

Arvo [1] extended the standard ray tracing algorithm by introducing a preprocessing step in which caustics are computed. This preprocessing step uses backward ray tracing (also known as light ray tracing, illumination ray tracing and photon tracing) in which packets of energy (photons) are emitted from the light sources in the scene towards the specular surfaces. Each photon is reflected by the specular surfaces and stored on the Lambertian surfaces. The main problem with this approach is computing the intensity (radiance) of the caustics. This value depends upon the number of photons per surface area. Arvo solved this problem by using illumination maps which is an empty texture map divided into a large number of small area-elements. As photons hit a surface the energy is registered at the appropriate area-elements. In this way the caustics are created as textures on the Lambertian surfaces within the scene. A problem with this approach is the fact that a large number of photons must be used to eliminate noise in the caustics. Heckbert [7] introduced a method that adaptively subdivided the illumination map into area elements (rexes) with a size corresponding to the local density of the photon-hits. Chen et al. [3] and Collins [4] use a fixed illumination map. To eliminate noise they use different filter-kernels to spread the energy from each photon onto several area-elements. Jensen et al. [9] stored all photon-hits explicitly in a photon map and avoided using the illumination map. Instead they introduced a new technique for estimating the number of photons per area by looking only on the distribution of photons within the scene. Their method is capable of handling complex objects (ie. procedurally defined objects) and the estimate is less prone to noise since it can be seen as a low-pass filter. The photon map does however require large amounts of memory in complex scenes.

In scenes with simple objects it is possible to avoid the photon based approach. If the specular objects are polyhedral backwards beam tracing [14, 15, 21] can be used to render caustics on the Lambertian surfaces. With backwards beam tracing the illumination map can be replaced by caustic polygons that represent illumination from caustics on Lambertian surfaces.

In bidirectional path tracing [11, 18] the rendering of caustics is significantly improved compared to traditional path tracing [8]. The method is however still purely stochastic and it still requires a large amount of sample rays to produce results that are not too noisy.

The most popular techniques today are clearly the backward ray tracing techniques as introduced by Arvo. These methods are often faster and more general than other approaches and they are often used in global illumination techniques [3, 9, 16] to render caustics. They do unfortunately have one significant drawback - they are limited to Lambertian surfaces. In many situations this is not a problem. However, within global illumination where accurate rendering is important the Lambertian assumption does not always produces satisfying results. It is however difficult to eliminate the Lambertian assumption since it removes the view independence of the caustics and therefore complicates the storage of irradiance on the surfaces.

In this paper we present a technique in which we extend the photon map in order to store irradiance on surfaces with reflection functions that are non-Lambertian. We achieve this by extending the information stored with each photon with the incoming direction of the photon. This allows us to combine the photons with general bidirectional reflectance distribution functions. We present results which demonstrate rendering of caustics on surfaces with reflection functions ranging from Lambertian to almost glossy specular.

2 The Photon Map

The photon map represents a rough distribution of light throughout the scene. It is created by emitting a large number of photons from the light sources into the scene. In [9] the photon map is used not only to simulate caustics but all kinds of illumination. We are only interested in caustics and we therefore construct a *caustics* photon map specifically aimed at rendering caustics.

The caustics photon map is constructed by emitting a large number of photons towards all the specular objects within the scene. Each time a photon hits a surface two things happen. If the surface is diffuse the photon is stored in the photon map, and if the surface has a specular component Russian roulette is used to determine whether the photon should be reflected specularly or absorbed. In this way we obtain an unbiased solution without

have to trace each photon through an infinite number of specular reflections.

Every photon is stored within the photon map. As [9] we use a kd-tree [2] to store the photons. While rendering the scene we need a data-structure that allows us to quickly locate photons within a given volume. Furthermore we need a very compact data-structure since we want to be able to use millions of photons. This makes the kd-tree a natural choice. In [9] the kd-tree was build on the fly as photons intersected the surfaces within the scene. This strategy can very easily result in a skew kd-tree that no longer has optimal search times. Searching is performed very often during rendering and we have found that balancing the kd-tree before actually rendering the scene significantly reduces the rendering time in most scenes. The balancing algorithm is performed after all the photons have been emitted. The photons are stored in a linked list of large arrays (each array having 65536 photons). The balancing algorithm manipulates this data structure directly in order to avoid having two copies of the photon map in memory. The balancing algorithm converts the unordered list of photons into a balanced kd-tree by recursively selecting the root node among the data-set as the median element in the direction which represents the largest interval. The existing data structure can be reused since we use a heap structure to represent the balanced tree. This completely eliminates the need for child-pointers. Further information on how to balance kd-trees can be found in [2].

As mentioned a large number of photons might be used in the photon map and it is necessary to use a compact representation. We have decided to use the following representation in which each photon only uses 20 bytes:

```
struct photon {
  float position[3];
  rgbe energy;
  char theta,phi; // incoming direction
  short flags;
  }
```

This representation is actually more compact than the one presented in [9] even though we have added information about the incoming direction. The use of a heap-like data-structure eliminates the need of two child pointers which would otherwise increase the memory requirements for each photon with 8 bytes (40%). The energy is represented as 3 floats packed into 4 bytes using the technique described in [19].

3 Rendering Caustics with the Photon Map

In standard ray tracing diffuse surfaces are only illuminated by the light sources. By introducing the photon map we have photons representing energy from caustics deposited on these diffuse surfaces. To render the caustics we need to extract radiance information from the photon map. This means that we must compute the density of the photons on all area-elements within the scene. Assuming that we have an intersection point $\mathbf{x}$ with normal $\vec{n}$ and an outgoing direction Ψ_r in which we want to compute the radiance, L_r, by using the photon map. L_r can be expressed as

$$L_r(\mathbf{x}, \Psi_r) = \int_{\text{all } \Psi_i} f_r(\mathbf{x}, \Psi_r, \Psi_i) L_i(\mathbf{x}, \Psi_i) |\vec{n} \cdot \Psi_i| \, d\omega_i \tag{1}$$

where L_i is the incoming radiance from the direction Ψ_i, and f_r is the bidirectional reflectance distribution function.

To compute the contribution L_i we locate the N photons with the shortest distance to $\mathbf{x}$. If we assume that each photon p represents a packet of energy (flux) $\Delta\Phi_p$ arriving at $\mathbf{x}$ from direction $\Psi_{i,p}$ then it is possible to integrate the information into equation 1 as follows

$$\begin{aligned} L_r(\mathbf{x}, \Psi_r) &= \int_{\text{all } \Psi_i} f_r(\mathbf{x}, \Psi_r, \Psi_i) \frac{d^2\Phi_i(\mathbf{x}, \Psi_i)}{dA \, d\omega_i} \, d\omega_i \\ &\approx \sum_{p=1}^{N} f_r(\mathbf{x}, \Psi_r, \Psi_{i,p}) \frac{\Delta\Phi_p(\mathbf{x}, \Psi_{i,p})}{\Delta A} \end{aligned} \tag{2}$$

We use the same approximation of ΔA as [9]. That is we take a sphere centered at $\mathbf{x}$ and expand it until it contains N photons and has radius r. ΔA is then approximated as

$$\Delta A = \pi r^2 \tag{3}$$

and we can rewrite equation 2 as

$$L_r(\mathbf{x}, \Psi_r) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(\mathbf{x}, \Psi_r, \Psi_{i,p}) \Delta\Phi_p(\mathbf{x}, \Psi_{i,p}) \tag{4}$$

4 A Non-Lambertian Reflection Model

In order to test the algorithm we need a reflection model capable of simulating non-Lambertian reflection. Several comprehensive models exists for the purpose of accurately simulating the physical behavior of different materials. However we are only interested in a simple model that can be used to validate our algorithm. A suitable model was presented by Schlick in [13]. This model is simple and it has the very nice property that it provides a continuous transition from Lambertian reflection to glossy specular reflection. We omit the usage of anisotropic reflection (even though nothing in our model prevents us from simulating anisotropy) and use the following BRDF:

$$f_r = \frac{1 - G(v)G(v')}{\pi} + \frac{G(v)G(v')}{4\pi v v'} \left(\frac{\alpha}{(1 + \alpha t^2 - t^2)^2} \right) \tag{5}$$

where the function G represents a geometrical self-shadowing factor:

$$G(v) = \frac{v}{\alpha - \alpha v + v} \tag{6}$$

and $t = \frac{\vec{n} \cdot (\Psi_r + \Psi_i)}{|\vec{n} \cdot (\Psi_r + \Psi_i)|}$, $v = \vec{n} \cdot \Psi_r$, $v' = \vec{n} \cdot \Psi_i$ and α is the diffuse-specular factor varying from 1 (diffuse) to 0 (specular).

To test our algorithm with this reflection model we only have to modify one parameter, α. This value determines whether the surface is diffuse or specular. Our results in the following section refers to this value as the diffuse-specular component.

5 Results and Discussion

We have implemented and tested our rendering algorithm on a Silicon Graphics Onyx computer with 1GB RAM. Since our representation of the photon map is quite memory efficient we never needed the 1GB memory. In general we rendered caustics using approx. 5-10MB for the photon map. Only in extreme cases where the caustic is rendered on surfaces with a reflection function approaching glossy specular did we need a large number of photons. In these cases we used approx. 10-30MB of memory for the photon map.

Our first test case is shown in figure 1. This is the standard model used to illustrate caustics. The cardioid-shaped caustic is formed by placing a light source on the edge of a cylinder which has a reflective inner side.

The incoming direction of the light at the edge of the cardioid equals the tangent to the cardioid. This information is quite useful when we remove the Lambertian assumption from the receiving surface. It allows us to predict how the caustic should look as the surface becomes more glossy. Figure 1 contains 4 rendered images showing how the caustic looks as we change the diffuse-specular component of the surface from 1 to 0.01. As expected the intensity of the caustic is reduced mostly in those parts where the incoming direction of the light differs mostly from the incoming direction of the viewing ray. We used approx. 340.000 photons in all the images corresponding to 7 MB of memory. The quality of the images can be improved slightly by using more photons. The images have been rendered in 320x240 with 4 samples per pixel and the rendering time for the images was (from left to right) 22, 24, 27 and 42 seconds - just ray tracing the images takes 7 seconds. The rendering time increases as the surface becomes more glossy and the only reason for this is the fact that the image sampling algorithm requires more rays to render the caustic on the glossy surface. Using a fixed number of samples per pixel would make the rendering time the same for all the images.

Our second test case (figure 2) is a simple scene demonstrating what happens with the caustic from a glass sphere as the receiving surfaces become glossy. As we can see the shape of the caustics is no longer oval but curved. In this scene we had to use approx. 250.000 photons to obtain a nice caustic — using fewer photons makes the caustic look more blurred. The image was rendered in 640x480 with 4 samples per pixel and the rendering time was 182 seconds.

Our third test case (figure 3) is a more complex scene in which we benefited from usage of a non-Lambertian reflection model. It is a glass of cognac on a sand-surface. The sand is a fractal surface (with $2 \cdot 1024^2$ triangles) on which we have produced a synthetic sand-texture. We have used a diffuse-specular factor of 0.6 - using a Lambertian approximation makes the sand look more unnatural and flat. The caustic in this image was rendered using approx. 350.000 photons. The image was rendered in 26 minutes in the resolution 640x480 with 4 samples per pixel. Notice how the red-looking caustic is formed as light is transmitted through several layers of glass and cognac. The intensity of each photon is modified using Beer's law as the photon is transmitted through a dielectric media.

In the following table we have collected some statistics of the resources required to render the images:

Image	Photons	Preprocess	Rendering
Figure 1[a]	336.191	8 min.	22-42 s.
Figure 2	250.677	58 s.	182 s.
Figure 2[b]	250.677	58 s.	378 s.
Figure 2[c]	5.036.126	19 min.	276 s.
Figure 2[d]	5.036.126	19 min.	1380 s.
Figure 3	352.497	15 min.	26 min.
Figure 3[e]	352.497	15 min.	42 min.

[a] Applies to all images in figure 1

[b] Without balancing the photon map

[c] A reference image demonstrating how the rendering time is (un)affected by the number of photons used

[d] The same as 2[c] but with an unbalanced photon map

[e] Without balancing the photon map

As shown in the table we also rendered figure 2 and figure 3 without balancing the photon map and this clearly affected the rendering times in particular as the number of photons increased. The rendering times were almost doubled with the unbalanced version. We also examined how the rendering times were affected as more photons were added and we rendered figure 2 using 5.0 million photons corresponding to a data-structure of almost 100 MB. Naturally this increased the preprocessing time due to the extra photons emitted from the light source. The time used in the balancing algorithm were less than a minute. As we can see from the table the balancing algorithm reduces the rendering time with more than 75 % In general we have noticed that rendering time with the balanced photon map is only slightly affected as the number of photons is increased. This is particularly important in situations where high quality is required or in situations where large parts of the scene are illuminated by caustics. It also makes the photon map easier to use since the primary parameter becomes the amount of memory available.

Currently the user must specify both how many photons should be generated at the light sources and how many photons N to use in the radiance computation (equation 4). It would be nice to have an adaptive method that based upon the local density of the photons determined how many photons to use in the estimate. In general we have found that it is quite easy to predict good values for the two parameters. If the parameters are badly chosen

the caustic will either become too blurred or too noisy.

The computed caustics are completely view-independent (even image independent). We do not need an initial ray tracing pass to determine the "bucket-size" as the illumination map based approaches. This also means that if very complex caustics are being visualized the user needs to adjust the number of photons used according to the desired resolution of the display. Another solution is just to always use enough photons if the memory permits it. As we have shown balancing the photon map (kd-tree) almost eliminates the dependence of the rendering time on the number of photons.

The rendering times could also be reduced even more by optimizing the integration of the photon map with Schlick's reflection model. Since we only have a discrete set of directions we could benefit from lookup tables and save a lot of vector computations.

The next step is integration of the method into a global illumination algorithm and extending the use of the photon map to other kinds of indirect illumination as in [9].

6 Conclusion

We have presented a new algorithm for rendering caustics on non-Lambertian surfaces. The method is based on an extension of the photon map and it renders caustics on procedurally defined surfaces. By balancing the photon map data structure we improve the rendering time and reduce memory requirements. The resulting method is fast and general and our test-images demonstrate that it is possible to achieve good results using only a limited amount of photons. The method is therefore useful in existing global illumination techniques in which caustics can be computed separately.

7 Acknowledgment

Thanks to Niels Jørgen Christensen, the reviewers and to Per Christensen and Martin Grabenstein for their helpful comments.

References

[1] Arvo, James: "Backward Ray Tracing". *Developments in Ray Tracing. ACM Siggraph Course Notes* **12**, pp. 259-263, 1986

[2] Bentley, Jon Louis: "Multidimensional Binary Search Trees Used for Associative Searching". *Comm. of the ACM* **18** (9), pp. 509-517, 1975

[3] Chen, Eric Shenchang; Holly E. Rushmeier, Gavin Miller and Douglass Turner: "A Progressive Multi-Pass Method for Global Illumination". *Computer Graphics* **25** (4), pp. 164-174, 1991

[4] Collins, Steven: "Adaptive Splatting for Specular to Diffuse Light Transport". In *proceedings of 5. Eurographics Workshop on Rendering*, pp. 119-135, Darmstadt 1994

[5] Cook, Robert L.: "Distributed Ray Tracing". *Computer Graphics* **18** (3), pp. 137-145, 1984

[6] Glassner, Andrew S.: "Principles of Digital Images Synthesis". *Morgan Kaufmann Publishers Inc.* 1995.

[7] Heckbert, Paul S.: "Adaptive Radiosity Textures for Bidirectional Ray Tracing". *Computer Graphics* **24** (4), pp. 145-154, 1990

[8] Kajiya, James T.: "The Rendering Equation". *Computer Graphics* **20** (4), pp. 143-149, 1986

[9] Jensen, Henrik Wann and Niels Jørgen Christensen: "Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects". *Computers and Graphics* **19** (2), pp. 215-224, 1995

[10] Jensen, Henrik Wann: "Importance Driven Path Tracing using the Photon Map". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 326-335, 1995

[11] Lafortune, Eric P.; Yves D. Willems: "Bidirectional Path Tracing". *Proceedings of CompuGraphics*, pp. 95-104, 1993

[12] Mitchell, Don and Pat Hanrahan: "Illumination from Curved Reflectors". *Computer Graphics* **26** (4), pp. 283-291, 1992

[13] Schlick, Christophe: "A Customizable Reflectance Model for Everyday Rendering". In *proceedings of 4. Eurographics Workshop on Rendering*, pp. 73-84, Paris 1993

[14] Shinya, Mikio; Tokiichiro Takahashi and Seiichiro Naito: "Principles and Applications of Pencil Tracing". *Computer Graphics* **21** (4), pp. 45-54, 1987

[15] Shinya, Mikio; Takafumi Saito and Tokiichiro Takahashi: "Rendering Techniques for Transparent Objects". *Proceedings of Graphics Interface '89*, pp. 173-182, 1989

[16] Shirley, Peter: "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes". *Proceedings of Graphics Interface '90*, pp. 205-212, 1990

[17] Shirley, Peter; Bretton Wade; Phillip Hubbard; David Zareski; Bruce Walter and Donald P. Greenberg: "Global Illumination via Density Estimation". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 219-230, 1995

[18] Veach, Eric and Leonidas Guibas: "Bidirectional Estimators for Light Transport". In *Proceedings of 5. Eurographics Workshop on Rendering*, pp. 147-162, Darmstadt 1994

[19] Ward, Greg: "Real pixels". In *Graphics Gems II*, James Arvo (ed.), *Academic Press*, pp. 80-83, 1991

[20] Ward, Greg: "The RADIANCE Lighting Simulation System". In *Global Illumination*. ACM Siggraph Course Notes **18**, 1992

[21] Watt, Mark: "Light-Water Interaction using Backward Beam Tracing". *Computer Graphics* **24** (4), pp. 377-385, 1990

[22] Whitted, Turner: "An Improved Illumination Model for Computer Graphics". *Comm. of the ACM* **23** (6), pp. 343-349, 1980.

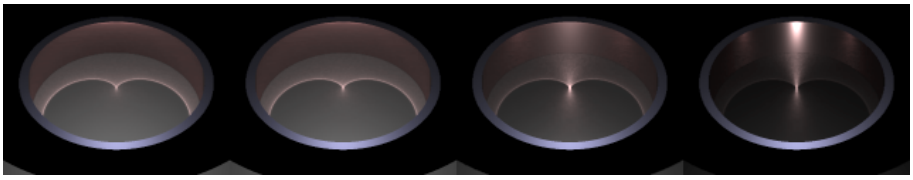

Figure 1: Four images demonstrating the looks of the cardioid created as light is reflected inside a cylinder-ring. From left to right the diffuse-specular component α is 1.0, 0.5, 0.1 and 0.01.

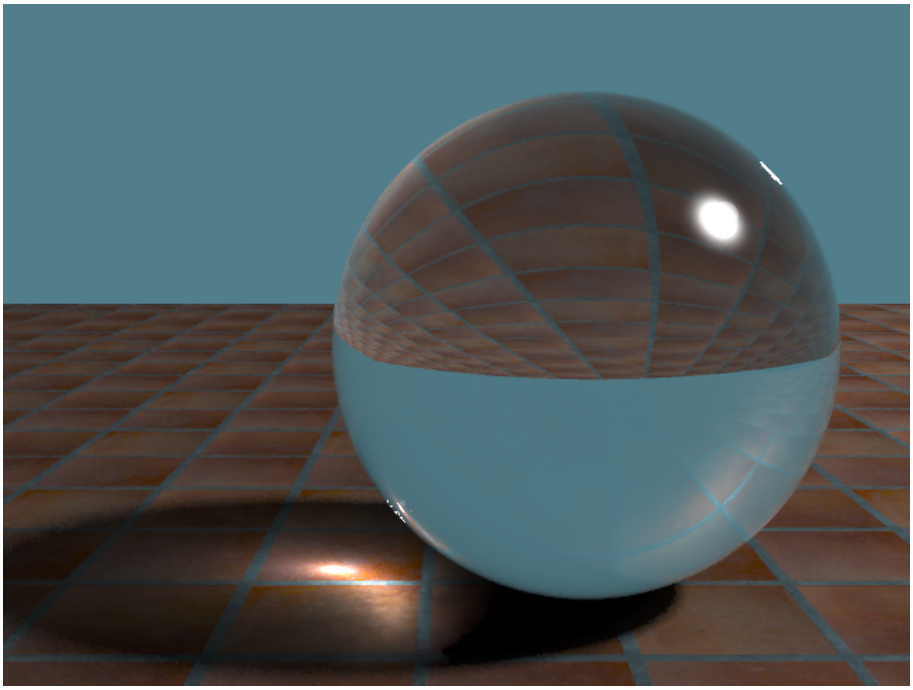

Figure 2: A caustic from a glass sphere onto a glossy stone surface with a diffuse-specular component $\alpha = 0.1$. Notice how the caustic becomes curved instead of oval as it would on a Lambertian surface.

Figure 3: A glass of cognac on a sand-surface. The sand is a fractal surface with a synthetic sand-texture. The diffuse-specular component of the surface is $\alpha = 0.6$ and this value improves the realism of the sand compared to a Lambertian approximation.