Beyond Faster Photon Map Global Illumination

by

Daniel Neilson
and
Yee-Hong Yang

Technical Report TR 04-21
October 2004

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

Abstract

Precalculating irradiance for photon maps is an enhancement to the photon mapping method that was proposed by Christensen to reduce redundant irradiance calculations during rendering. In this enhancement, irradiance values are precalculated at the location of photons in the photon map prior to rendering and are stored within the photon's data structure. These precalculated irradiance values are then used during rendering. However, this enhancement may calculate irradiance values that are not used during rendering. This paper presents an improvement to Christensen's method, called on-demand caching, that removes these unused irradiance calculations by performing irradiance calculations during rendering and caching the results in the photon map.

1 Introduction

Global illumination is very important when synthetically generating photorealistic images. As such, it is desirable to have global illumination algorithms that are as efficient as possible to allow for more complex and detailed scenes to be generated with the same resources.

The photon mapping method is a technique that can be used to calculate global illumination, including caustics, in a manner that is not tied to the geometry in the scene. This independence of the photon map from the geometry in the scene has made the method very popular within the graphics community. In an effort to improve the efficiency of the photon mapping method, Christensen [2] devised an improvement to the method that involves precalculating the irradiance in the scene prior to rendering and using these precalculated results during rendering. However, when using this improvement it is possible that irradiance values that are not needed during rendering will be calculated.

To remedy this inefficiency, we propose an improvement to Christensen's irradiance precalculation method that delays irradiance precalculation to the rendering stage of the photon mapping method. By delaying irradiance calculation to the rendering stage we are able to only calculate irradiance values that will be used while still allowing those values to be cached for later use. We show that this improvement can provide a speedup of 1.2 to 2.2 times in irradiance calculation, and will perform no more calculations than Christensen's irradiance precalculation method in the worst case.

2 Background

In this section, we briefly review the photon mapping method for computing global illumination as well as an original improvement proposed by Christensen [2] to speed up the method.

2.1 The Photon Map Method

The photon map is an implementation of the backward ray tracing method [1] that has seen increased popularity in recent years. The photon map method divides image generation into two phases; construction of two illumination maps (one for global illumination, and one for caustics), and rendering.

The illumination map is constructed by emitting light photons from the light sources in the scene and tracing them using Monte Carlo techniques [4]. Based on the properties of the material that a photon hits, the photon will either be reflected, transmitted, or absorbed (note that absorption only occurs in diffuse materials). When a photon is absorbed, its position, energy, incident direction, and a few flags are stored into an illumination map; if the photon is coming from a specular object then it is stored in the caustics map, otherwise it is stored in the global illumination map). Typically, this requires 18.25 bytes of storage per photon [4] and thus each photon is stored in a 19 byte structure in which 6 bits are unused. Once all photons have been emitted, the illumination map is sorted into a kd-tree structure to speed up illumination calculations.

In the rendering phase, standard ray tracing techniques are used with a difference only in the method used to calculate incident illumination on diffuse surfaces. If the contribution of the ray to the rendered image is low, then the illumination is calculated from the power and density of the photons nearest to the sample. On the other hand, when the ray's contribution to the rendered image is high, the incident illumination is calculated by combining direct illumination, caustics computed from the caustics illumination map, and soft indirect illumination computed with final gathering using the global illumination map, which is simply a single level of distributed ray tracing [3] in which all illumination is calculated via the illumination map.

2.2 Precomputing Irradiance

Christensen [2] proposed a method for improving the efficiency of the final gathering portion of the photon map method by a factor of 5 to 7. He observed that rays from different final gathering calculations compute the irradiance repeatedly at nearly the same location. Thus, he proposed precalculating the irradiance at each photon location and storing this precalculated irradiance value within the photon structure. Along with the precomputed irradiance, Christensen also stores the surface normal at the photon location; this normal is used when looking up an irradiance value during final gathering. In all, Christensen's proposed method increases the size of a photon by 5 bytes to 24 bytes (6 bits of which are unused); 1 byte is added to store an encoding of the surface normal, and 4 bytes are added for the irradiance using Ward's shared exponent representation [6].

During final gathering, when an irradiance value is needed at a point $\boldsymbol{x}$ the nearest photon with a similar normal to the surface normal at $\boldsymbol{x}$ is found. The precomputed irradiance stored in this photon is then used as the irradiance at $\boldsymbol{x}$.

3 On-demand Caching of Irradiance Values

The irradiance precalculation method for photon maps proposed by Christensen [2] blindly calculates the irradiance at all photon locations, regardless of whether or not a given calculation will actually be used while rendering the image. If a large number of these precalculated irradiance values are not used while rendering the image, then performance of the ray tracer will suffer. In this section, a modification to Christensen's method, called on-demand irradiance caching, is presented that eliminates calculating irradiance values that are not used to render the image.

Rather than precalculating irradiance at every photon location, irradiance is calculated only as it is needed. To do this, a 1-bit flag is added to each photon that records whether or not the irradiance has been calculated at the photon location. Fortunately, since there are 6 bits of unused space in each photon, adding this flag does not increase the size of a photon.

During final gathering, the irradiance at the point $\boldsymbol{x}$ is calculated by finding the nearest photon in exactly the same manner as in Christensen's method. The flag is then checked, and if it indicates that the irradiance at the photon's location has not yet been calculated then the irradiance is calculated at the photon's location. The calculated irradiance is then stored in the photon, and the flag is set to indicate that the irradiance has been calculated. The calculated irradiance stored in the photon is then used as the irradiance at $\boldsymbol{x}$.

By modifying Christensen's method in this manner, we no longer calculate any irradiance values that will not be used during rendering. Thus, in the worst case we would calculate the same number of irradiance values as in Christensen's method. But, we expect that, on average, the reduction in irradiance calculations performed will result in a noticeable speed-up. Furthermore, Christensen's method offers no speed-up (in fact it slows down rendering) if the number of irradiance calculations used in final gathering is less than the number of photons in the illumination maps. By making the modification proposed in this paper this restriction on the applicability of the method is removed; since irradiance values are only calculated as they are required, there will be no slow down if the number of irradiance values needed is less than the number of photons.

4 Results

To test the effectiveness of the on-demand irradiance caching enhancement we constructed a test suite of 40 scenes which were rendered using both irradiance precalculation and on-demand irradiance caching. Each scene in the test suite is of a $40 \times 40 \times 40$ enclosed box, with a 5×5 rectangular area light at the top, and containing a variable number of diffuse spheres. The spheres in each scene are generated such that scene one contains one randomly generated sphere, and scene $i + 1$ contains the same spheres as scene i plus one additional randomly generated sphere; each sphere was generated with a radius in the range $[3, 5]$.

Each of the scenes in the test suite was rendered to a 400×400 pixel image on a Pentium 4 2.26 GHz processor running Redhat Linux 9.0 (kernel version 2.4.20-20.9) and version 3.3.2 of the gcc compiler suite. The photon map for each scene contained 500,000 photons with the nearest 300 photons within a maximum radius of 3.0 being used for each irradiance calculation.

Figure 1 shows the total time taken to ray trace each of the 40 scenes in the test suite using only indirect illumination calculated via the photon map for illumination. By not calculating any direct lighting for these images we are able to focus on only the portion

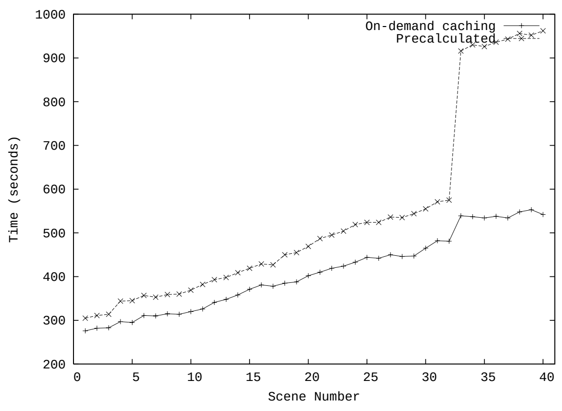

Figure 1: Time (in seconds) spent ray tracing the 40 test scenes with Christensen's irradiance precalculation method and the new on-demand irradiance caching method.

of ray tracing that is relevant to the photon map; namely photon map construction, irradiance precalculation (when applicable), and irradiance estimation via the photon map. By using the on-demand irradiance caching method, this portion of the ray tracing required anywhere from 46% to 84% of the time that was required when precalculating irradiance. As seen in figure 2, this corresponds to a speedup from 1.2 to 2.2 times.

The large jump in the time required by the irradiance precalculating method at scene 33 corresponds with the addition of a single large sphere very near to the light source. The addition of this sphere causes some areas of the scene, that are not reached very often during final gathering, to contain a very low photon density. Since the irradiance precalculating method precalculates irradiance at all photon locations, regardless of whether a given irradiance value will ever be used, the method ends up calculating irradiance throughout these low photon-density regions resulting in a substantial slow down in generating the image.

5 Conclusions

In this paper we have addressed a problem with Christensen's precomputation of irradiance for photon maps that results in irradiance values being calculated even when they might not be used in ray tracing. We present an enhancement that has been shown to speed up Christensen's method by 1.2 to 2.2 times. In the worst case, the enhancement performs identical to Christensen's method of irradiance precalculation.

6 Acknowledgments

The authors would like to thank the generous funding of: NSERC, the NSERC PGS-A Postgraduate Scholarship Program, the Alberta Informatics Circle of Research Excel-

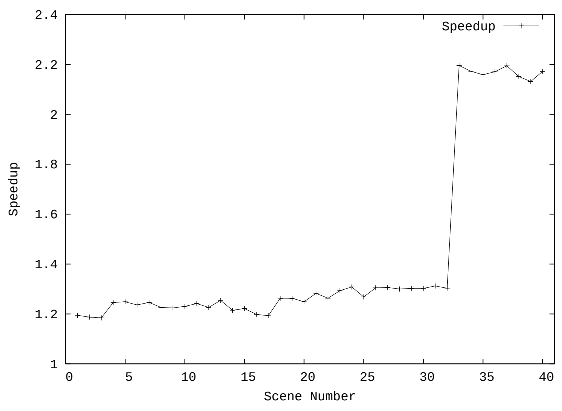

Figure 2: Speedup in the 40 test scenes for using the new on-demand irradiance caching method as opposed to Christensen's irradiance precalculation method.

lence's (iCORE) Graduate Student Scholarship Program, the Department of Computing Science, and the University of Alberta.

References

[1] J. Arvo. Backwards ray tracing. In *Developments in Ray Tracing*, volume 12 of *SIGGRAPH Course Notes*, 1986.

[2] P. H. Christensen. Faster photon map global illumination. *Journal of Graphics Tools*, 4(3):1–10, 1999.

[3] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *ACM SIGGRAPH*, pages 137–145, 1984.

[4] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A.K. Peters, 2001.

[5] D. Neilson. Efficient algorithms in motion blurring and photon mapping. Master's thesis, University of Alberta, August 2003.

[6] G. Ward. Real pixels. In J. Arvo, editor, *Graphics Gems II*, pages 80–83. Academic Press, 1991.