Interactive Global Illumination for Quasi-Static Scenes

R. J. García, C. Ureña, M. Lastra, R. Montes and J. Revelles
Departamento de Lenguajes y Sistemas Informáticos. University of Granada. Granada, Spain.
{ruben,almagro,mlastral,rosana,jrevelle}@ugr.es

Abstract

This paper describes an approach to obtain interactive recalculation of global illumination for scenes with small moving objects (with respect to the complete geometry), on a standard PC, using Density Estimation techniques.

1 Introduction and State of the Art

1.1 Introduction

The purpose of this article is describing an approach to obtain interactive global illumination in a quasi-static scenes, using a desktop PC and a standard compiler. A quasi-static scene is defined as the combination of a static scenario and a dynamic object, when the size of the dynamic object is much smaller than the complete scenario.

In the approach presented in this article, the dynamic object refers to the object or set of objects moving in the frame being calculated. This means that it can also be used for scenes in which most or all of the objects move along the animation, as long as only a small coherent fraction of them (i.e. a few solids) moves in each frame. This is the most usual scene type, especially in videogames, which are one of the most common interactive graphical applications.

1.2 Density Estimation Methods

In order to obtain a radiosity value in a point, the integral of the incident radiance for all the directions in that point is calculated. One method which has proven useful to obtain an estimate of the integral is the Density Estimation Method, popularized by [14]. This method consists of two phases. The first phase is based on the particle model of light, and traces a number of photons from the light sources. The second phase (Density Estimation proper) estimates the radiance. [14] mentions a third phase, decimation, which simplifies geometry after illumination has been calculated.

The most basic method is described by Arvo in [1] and Patanaik in [11] and is refered to as Impact Count (or IC) in this article. It finds the impacts of the photons in the triangles and calculates the energy density in the triangle. Then vertexes have a radiance which is the average of the triangles to which they belong.

Henrik van Jensen [7], devised the well known Photon Maps (PM) method. It consists in finding the nearest n photons (n is predefined) of the point where radiance is being estimated, adding their energy, and dividing by the area of the greatest circle of the sphere which contains the n photons. This method's most known limitation is that when radiance on a point is calculated, nearby photon impacts should lie on the same plane and be in a circular area arond the point. [6] presents an algorithm which solves this limitation by using geometry information near the point.

Another less known limitation of Photon Maps and [6] is that if relativelly very small surfaces exist in the scene, these zones have a comparately very high variance, and they tend to appear either too bright or too dark if the number of photons is not large enough.

A method to avoid this high variance consists in storing the rays in the scene and using a fixed size disc centered in the point where radiance is being calculated, and contained in the plane tangent to the surface. Rays intersecting a given disc are used to calculate the radiance in the point on which the disc is centered [10][9]. The algorithm is called Density Estimation on the Tangent Plane (DETP).

1.3 Interactive Global Illumination

There are some algorithms to calculate interactive global illumination. Finite Element Methods [3] were the first ones developed for the calculation of radiosity and radiance. They divide the scene in a set of patches, and calculate the form factor between pairs of patches. Then a system of equations is solved in order to know the radiance in each patch.

Wald et al, in [13], explain a method which applies techniques from Distributed Programming to RayTracing. It uses a cluster of PCs to obtain global illumination in real time.

Benthin [2] describes the changes in the system to obtain improved scalability and performance.

Dmitriev, in [4] proposes a method which allows for the recalculation of illumination and radiosity along a number

Proceedings of the Computer Graphics International (CGI'04)
1530-1052/04 $20.00 © 2004 **IEEE**

of frames, and for the fast finding of the photons which need to be recalculated. He uses raytracing from the light sources, so his solution is view-independent.

2 Algorith Overview

2.1 Motivation

All the algorithms proposed in the previous section have some drawbacks when applied to rendering of quasi-static scenes in Real-Time. The Density Estimation algorithms present in section 1.2 were designed for the rendering of a still image. Therefore, they do not reuse information from the previous frame when used to compute an animation.

Finite element methods have the limitation that the movement of one patch affects the form factor of the patch with all the other patches.

The algorithm Wald et al [13] proposes is based on a Real-Time ray-tracing system, but has the limitations that it cannot be mapped to an architecture of a standard PC, and there is a delay in the illumination.

The problem is analogous to the one present in Finite Element Methods.

However we think the usage of Wald et al's Real Time system would improve every method based on photon tracing which presents ray coherence, as is the one presented here.

2.2 Simplifications

In order to obtain an interactive algorithm, some simplifications were a reasonable trade-off between functionality and performance: The only allowed primitives are triangles, lights should be static, the scene should be quasi-static, and surfaces should be opaque and diffuse.

2.3 Basics

Our algorithm is based in the density estimation techniques. For the first frame, a photosimulation phase is performed, and then a density estimation technique is applied. For latter frames, radiance is recalculated by an incremental update of the information gathered in the above phases.

2.4 Computing and updating the ray set

The algorithm's first step is the generation of the rays and storage of the intersection points of each ray with the scene.

In latter frames, the ray set is updated this way: The triangle intersected by the ray in the previous frame is known (it is stored in the ray). If this triangle belongs to the mobile object (i.e. its position has changed), the ray is recalculated, taking into account the new position of the mobile object. If the triangle does not belong to the mobile object (or if there was no hit at all in the previous frame) the ray is tested against the mobile object's current position. If it hits, its reflections are recalculated. Otherwise, the ray is still valid, but the procedure should be repeated for its reflections. Note that for most rays, no intersection test against the scene is needed, which is where most of the complexity lies.

In order to implement this approach efficiently, a two-level ray list is used: The first level contains the primary rays; these rays allow access to a simply linked list of rays produced by subsequent reflections of the photons. Recalculating a ray means storing a copy of the ray (drawing along a reference to the rest of the reflections of the photon) in a helper list, referred to as "Old Rays" from now on, and rechecking the original ray against the scene and mobile object. This will generate a list of new reflections, which are stored where the reference to the old reflections was. They also have to be stored in another helper list, "New Rays". With this approach, there are three ray lists: Old, New and Current Rays. These lists are later used to calculate radiosity values in the vertexes.

2.5 Computing and updating radiosity

Now we focus on the estimation of radiance at each frame, which must use that ray set as input data.

2.5.1 Calculation of the first frame

The first time the radiosity algorithm is run, all the rays in the scene must be taken into account when calculating the radiance of a vertex.

This algorithm's approach to the recalculation of irradiance requires that the points in which radiance is calculated be fixed along the simulation, and the vertexes were the most useful candidates.

2.5.2 Updating radiosity in static vertexes

On the next run of the algorithm, the mobile object is located in a new position. The rays which intersected the object in the previous or this frame have been recalculated in the Photosimulation phase. However, most of the rays have the same contribution as before to the radiosity on a given point.

Let $E(x,t)$ be the irradiance at x at instant t. Let $\tilde{E}(x;S)$ be the irradiance at x due to the set of rays S. Let S^t be the set of rays at instant t. We define $\tilde{E}(x,t)$ as the approximation to radiance on point x calculated from the rays on time t; formally:

$$E(x,t) \approx \tilde{E}(x,t) =_{def} \tilde{E}(x;S^t) \quad (1)$$

[5] demonstrates how the radiance can be updated: Let N^t be the new ray set at frame number t, and O^t the old ray set in that frame. For all the frames but the first (i.e. $\forall t > 0$), it holds that $N^t = S^t - S^{t-1}$ and $O^t = S^{t-1} - S^t$. If the density estimation technique is linear ($\forall S_1, S_2, x, S_1 \cap S_2 = \emptyset \rightarrow \tilde{E}(x;S_1 \cup S_2) = \tilde{E}(x;S_1) + \tilde{E}(x;S_2)$) radiosity can be updated by:

$$E(x,t) \approx \tilde{E}(x,t-1) + \tilde{E}(x;N^t) - \tilde{E}(x;O^t) \quad (2)$$

In order to recalculate the radiosity, the density estimation algorithm is run with the old ray set generated in the previous step. The calculated radiance is substracted to the previous value in each vertex. Then the algorithm is run again with the new ray set, and the calculated radiance estimate is added to the value in the vertex.

2.5.3 Computing and updating radiosity on dynamic vertexes

Dynamic vertexes are those which changed their position or orientation in the frame which is being computed.

The algorithm for dynamic vertexes depends on the density estimation method. The algorithms are presented next.

Impact Count

In order to update the radiosity using Impact Count, the algorithm needs to know which triangles have received new impacts or no longer have the impacts they had. Fortunately, the Photon tracking phase calculates this information, so the contribution of the rays which do not hit the triangles any longer can be substracted and the new impacts added. This can be done because each ray impacts with exactly one triangle, or leaves the scene. It is easy to keep track of the triangle hit in the ray.

The radiosity on a vertex is finally calculated as an average of the radiosity of the triangles it belongs to. Therefore, the algorithm for recalculation of the mobile objects with Impact Count is identical to the recalculation of the static scene.

Photon Maps and Density Estimation on the Tangent Plane

To calculate the radiosity estimate in the points of the mobile object, the initial algorithm is run, checking all the rays in the scene.

The algorithm used in Impact Count cannot be used for Photon Maps nor Density Estimation on the Tangent Plane because some of the rays which are neither Old nor New can change their contribution to the illumination of the mobile since the discs (DETP) and the spheres (PM) change their distance to the rays when they move.

2.6 DETP Specific Optimizations

Some optimizations have been described which increase performance or reduce the bias of the algorithm in certain situations. One of them is Artifact Control and another is Ray Cache, both explained in [10].

Artifact Control removes some artifacts present in the images DETP generates, due to the fact that points in concave surfaces can have unreachable zones in their corresponding circles. Two things must be taken into account:

- A ray contributes to a circle if the intersection is strictly before the second intersection with the scene.
- If a portion of the disc is not visible from the origin of the ray, the contribution of the ray to the disc must be divided by the area of the visible zone.

Ray Cache refers to an optimization of the ray-disc intersection: A list of spheres is built, starting with a sphere which bounds the scene, and subsequent smaller spheres until the radius is smaller than a threshold. The threshold is proportional to the disc radius of DETP. Each sphere has a data structure which contains the rays which intersect it. The inner sphere is built so that it contains the first disc; then only intersection of the disc against the rays in the sphere are needed. For the rest of the discs, the inner spheres are rebuilt from inner to outer if the disc is not completely inside the given sphere. New spheres are centered in the point the radiance is being calculated and keep the old radius of the sphere.

The performance of the Ray Cache depends strongly on the spatial coherence of discs in order to reuse the spheres. Lastra et al[10] presented an algorithm which calculated a reordering of the vertexes for increased spatial coherence for the ray-circle intersection test, which was called Point Ordering.

This order can be precalculated and reused in the case of Quasi-Static scenes, since the order will not change for the static vertexes, and dynamic vertexes move coherently.

3 Quantitative comparison among different density estimation methods

A scene with 72 000 triangles and a 500 triangle mobile object will be used in this section.

The methods based in photosimulation use a spatial indexing technique to increase performance. In this article, an octree with the traversal algorithm introduced in [12] was used.

3.1 Comparison between Impact Count and Density Estimation on the Tangent Plane

It is obvious that the algorithm for IC is intrinsically faster than the one for DETP, since only the recalculated rays have to be taken into account.

Although it has been noted in the literature that the parameters of noise and variance are much higher in IC than in PM ([8], page 52) or DETP ([9], section 2), the time for the whole calculation of irradiance was studied. In this case, however, what is measured is the time of recalculating the whole scene in PM or DETP, versus recalculating rays which intersect the mobile only in IC.

The reference image is created by using 100 million rays with the DETP estimation method (radius is 1 % of the scene's bounding box side).

Algorithm	RR	DE	Total	Error
DETP	1.18	1.64	2.82	12.36 %
IC	3.01	0.03	3.04	159.95 %

Table 1. Comparison between Density Estimation on the Tangent Plane and Impact Count

The error of an image is calculated in this paper by calculating the percentual luminance differences of the image and the reference image for each vertex in the scene and averaging them.

Table 1 shows the time in seconds for the recalculation of the rays (RR), the density estimation (DE), and the total time. Finally the error is shown. It can be seen that while the simulation time is similar (slightly lower for DETP), the error is an order of magnitude higher in IC.

3.2 Comparison between Density Estimation on the Tangent Plane and Photon Maps

In this section the resulting timing and error information is given. The timing data can be seen in tables 2 and 3 for Density Estimation on the Tangent Plane and Photon Maps, respectively. DE corresponds to the Density Estimation phase. KP are KiloPhotons.

Ray Cache and Point Ordering are used in the DETP. For 1 000 photons, 9 frames per second is obtained For 10 000 rays, the refresh rate is 2 frames per second.

PM gets higher error at the corresponding speed of DETP.

KP	P. Tracing		DE		Frame	Error
	Init	Fr.	Init	Fr.	Time	
1	0.11	0.01	4.54	0.01	0.02	32 %
5	0.53	0.02	27.76	0.07	0.09	18 %
10	1.01	0.04	73.14	0.16	0.20	15 %
50	5.33	0.18	589.23	1.48	1.66	14 %

Table 2. Density Estimation on the Tangent Plane

The reference files are an image generated with Photon Maps, 5 million photons, and two images with Density Estimation on the Tangent Plane, 5 million photons.

4 Future Work

Our lines of future work include: General reflective and refractive surface properties, Hardware accelerated direct illumination, Quasi Monte-Carlo, the multidimensional Halton sequence and use of hardware to calculate shadows[4].

KP	P. Tracing		DE		Frame	Error
	Init	Fr.	Init	Fr.	Time	
2	0.22	0	0.18	0	0	125 %
20	2.04	0.06	0.31	0.07	0.13	80 %
200	21.10	0.69	2.31	2.17	2.86	67 %

Table 3. Photon Maps

Acknowledgements

This work has been supported by the research project coded TIC2001-2392-C03-03 (Spanish Commission for Science and Technology), and by Grant AP2001-3238, from the FPU program of the Spanish Ministry of Education, Culture and Sport.

References

[1] J. R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, pages 259–263, 1986.

[2] C. Benthin, I. Wald, and P. Slusallek. A scalable approach to interactive global illumination. In *Computer Graphics Forum*, volume 22. Eurographics, Blackwell, September 2003.

[3] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Harcourt Brace & Company, 1993.

[4] K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel. Interactive global illumination using selective photon tracing. 13^{th} *Eurographics Workshop on Rendering*, 2002.

[5] R. J. García, C. Ureña, and M. Lastra. LSI-2003-1 Real Time Global Illumination for Quasi-Static Scenes. Technical report, Departamento de Lenguajes y Sistemas Informáticos (Software Engineering). University of Granada, 2003.

[6] H. Hey and W. Purgathofer. Advanced radiance estimation for photon map global illumination. *Eurographics 2002*, 2002.

[7] H. Jensen. Global illumination using photon maps. In Pueyo and Schroeder, editors, *Rendering Techniques'96*, pages 21–30. Springer-Verlag, 1996.

[8] H. W. Jensen. *Realistic Image Synthesis using Phothon Mapping*. AK Peters, 2001.

[9] M. Lastra, C. Ureña, J. Revelles, and R. Montes. A density estimation technique for radiosity. *1st Ibero-American Symposium in Computer Graphics (SIACG'2002)*, 2002.

[10] M. Lastra, C. Ureña, J. Revelles, and R. Montes. A particle-path based method for Monte-Carlo density estimation. *Poster at: 13th EUROGRAPHICS Workshop on Rendering*, 2002.

[11] S. Pattanaik and S. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. *Proceedings of 3rd Eurographics Rendering Workshop, Bristol*, 1992.

[12] J. Revelles, C. Ureña, and M. Lastra. An efficient parametric algorithm for octree traversal. *Journal of WSCG (UNION Agency-Science Press)*, 8(2):212–219, 2000.

[13] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. TR-2002-02: Interactive global illumination. Technical report, Computer Graphics Group, Saarland University, 2002.

[14] B. Walter, P. M. Hubbard, P. Shirley, and D. P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, 1997.