Volume xx (200y), Number z, pp. 1–7

Affichage non alias pour l'illumination globale par voxels

Łukasz Piwowar[1] and Rémy Malgouyres[2]

[1]Institute of Computer Science, University of Wrocław,Wrocław 50-383,
Joliot-Curie 15, Poland
lpi@ii.uni.wroc.pl
[2]Univ. Clermont 1,
Laboratoire d'Algorithmique et Image (LAIC, EA2146),
IUT dpartement informatique
B.P. 86, 63172 Aubière cedex, France
remy.malgouyres@laic.u-clermont1.fr

Abstract
Nous prsentons une mthode d'affichage non aliase base sur le lancer de rayons pour l'illumination globale par voxels, qui est un alogorithme non biais, introduit dans [Mal02], [CM06], [ZFM06], pour rsoudre l'quation d'illumination globale. Cette mthode hybride consiste reprsenter le champs de radiance entrant par des sources luminauses ponctuelles virtuelles (VPLs), en composant la solution initiale en utilisant une structure de donnes spaciale forme de voxels. Le principal avantage est l'utilisation de la solution linaire optimale pour rsoudre les problmes de visibilit avec cette structure de donnes de voxels.

1. Introduction

L'un des sujets aujourd'hui brlant en synthse d'images est l'illumination globale, et la recherche d'un moyen pour animer en temps rel un environnement virtuel photo-raliste. L'objectif de l'illumination globale est de gnrer des images ralistes de scnes $3D$, qui soient correctes quantitativement concerant l'nergie mise par les diffrents lments de la scne.

Les donnes en entre sont la description de la scne et de la camra, des maillages et les sources lumineuses courramment utilises en synthse d'images. Introduite par James Kajiya in 1986, l'quation d'illumination globale dcrit le transfer d'nergie lumineuse d"un point un autre comme une somme de la radiance mise et de la radiance rflchie. Il en dcoule une quation d'quilibre nergtique appele *quation d'illumination globale*. Les mthode d'illumination globale exactes tentent de calculer une solution numrique exacte pour cette quation. La solution ne dpend pas du point de vue.

Dans [Mal02], une approche entirement nouvelle est propose, ainsi qu'une discrtisation de l'quation d'illumination diffuse, base sur une approximation des surfaces par des voxels. Cette discrtisation de l'quation conduit un systme linaire dont les inconnues sont la radiance en chaque voxel. Une mthode pour rsoudre ce systme linaire est prsente qui est beaucoup trop chre pour tre praticable.

Dans [CM06], **une solution optimale en complexit (linaire par rapport au nombre de rayons) est obtenue pour rsoudre les problmes de visibilit**. C'est un atout trs fort pour cette approche comparativement au photon-maping ou a bidirectional path-tracing, qui sont fondes sur des algorithmes d'intersection rayon-objet. Cependant, l'affichage des voxels restait fortement alias et ncessitait de nombreux voxels pour produire un rendu acceptable, et la mthode parraissait toujours impraticable. Dans [ZFM06], on propose une version avec mmoire distribue pour les scnes ayant de nombreux voxels.

Dans cet article, nous produisons une nouvelle mthode d'affichage utilisant un lancer de rayons progressif. L'affichage est non alias et produit la fois des rsultats prcis et un bon rendu. Cel permet de rduire de beaucoup le nombre de voxels et le nombre de directions, faisant de notre mthode d'illumination globale, avec sa solution optimale unique en son genre pour la visibilit, un algorithme conptitif pour l'clairage indirect avec plusieurs rflexions successives. Comme seule la phase de lancer de rayons dpend du point de vue, nous avons de relles perspective d'animation temps rel de scnes statiques.

2. Illumination Equation and Classical Approaches

Let x and y denote two points in a scene. Let $B(x)$ denote the *radiosity* at the point x, the amount of the energy leaving x per unit of time per unit of solid angle (steradian) per unit of area. Let V denote the visibility function, defined by $V(x,y) = 1$ if y is visible from x in the scene and $V(x,y) = 0$ otherwise. Let r denote the distance between two points x and y. Let $\theta(x,y)$ denote the angle between the vector $\overrightarrow{xy}$ and the normal vector $\overrightarrow{n}$ at the point x (see Figure 1). If that angle is less then or equal to $\frac{\pi}{2}$, there is no interaction between x and y, and we define by convention $\cos\theta$ as equal to 0. The (continuous) diffuse illumination equation is as follows:

$$B(x) = E(x) + \rho(x) \int B(y) \frac{\cos\theta(x,y)\cos\theta(y,x)}{\pi r^2} V(x,y)\, dy$$

where

- A point re-emits only a fraction $\rho_d(x)$ of the energy it receives. Assuming that this factor does not depend on incoming or outgoing directions is known as the *ideal diffuse hypothesis*. The reflection coefficient ρ_d is less than 1.
- $\theta(y,x)$ is the angle between the vector $\overrightarrow{yx}$ and the normal $\overrightarrow{n'}$ to the surface at y.
- The π factor is a normalization term deriving from radiance considerations.

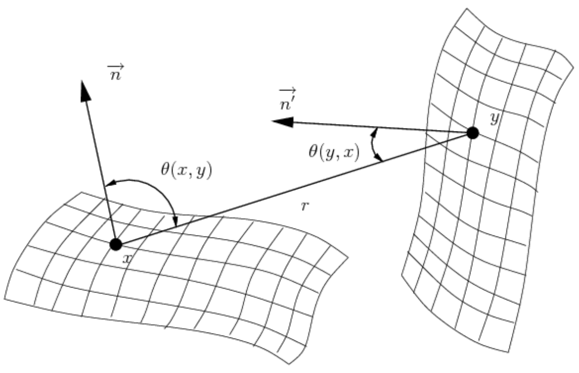


Figure 1: *Geometric relation*

An intuitive explanation of Equation (1) is: "the total power of light leaving a voxel x (the $B(x)$ term), is defined by two terms: the proper emittance of this voxel as a light source (the $E(x)$ term), and some fraction of the light it receives from its environment (the integral)".

Note that the equation presented here is a particular case, corresponding to the Lambertian model of reflection, of a more general global illumination equation with a *Bidirectional Reflectance Distribution Function* (*BRDF*). There is no theoretical obstruction to using our method with *BRDFs*, but only the diffuse case is fully implemented at this point.

We cannot present exhaustively all known approaches to global illumination in this paper for lack of space, but let us mention a few of them.

1. Radiosity with patches ([SP94]) decomposes the scene into some polygonal patches and makes the assumption that the radiosity is constant on each patch. The global illumination equation is then approximated by some linear system with good properties. The drawbacks of radiosity is the difficulty to make it work with *BRDFs*, and the high cost when the number of polygons increases (the visibility factor, called *form factor* between two patches is expensive to compute). Moreover, if we take too few patches, we cannot accurately display curved objects and mainly polygonal scenes are rendered.
2. Photon mapping ([Jen96], [Jen97], [Jen01]) is a method that traces random rays from light sources, and at each intersection, traces a new random ray with a probability that depends on the *BRDF*. This method accurately solves the illumination equation but indirect lighting is very expensive, especially since tracing a single ray, even with accelerating data structures, is far from constant time.
3. Bidirectional path tracing presents the same drawbacks as photon mapping.
4. Instant radiosity ([Kel97]) can be used in combination with other methods such as photon mapping for good looking display. Photons are used as virtual light sources in a raytracing phase. Instant radiosity is viewpoint-dependent. The method is very expensive on scenes lit primarily by indirect lighting, either with photon maps or quasi-random walk to distribute photons, several hundred frames may be required to get photons to arrive where they are needed.

Our method is from a family of methods that stores the outgoing energy in some spacial data structure (voxels in our case), which, similarly to radiosity, enables us to take advantage of dynamic programming, which is the only way to effectively simulate several successive reflections.

3. Voxel Based Approach

By a change of variable, using the correspondence between visible points from the point x and points on a sphere S centered at x, and the continuous diffuse illumination equation is equivalent to

$$B(x) = E(x) + \rho(x) \int_S B(y) \frac{\cos\theta}{\pi} d\overrightarrow{\sigma}$$

In [Mal02], we simply discretize this equation by approximating the surfaces by a discrete surface, and also sampling the sphere to estimate the integral, by introducing a finite set of directions.

$$B(x) = E(x) + \rho(x) \sum_{\overrightarrow{\sigma} \in D} B(I(x,\overrightarrow{\sigma})) \frac{\cos\theta(x, I(x,\overrightarrow{\sigma}))}{\pi} \Delta\Omega(\overrightarrow{\sigma}) \qquad (1)$$

where

- x is now a voxel;
- D is a set of discrete directions in space;
- $I(x, \overrightarrow{\sigma})$ is the first point y viewed from x in the direction of $\overrightarrow{\sigma}$ (as in a ray-object intersection);
- To quantify how much of an object is seen from a point, the term $\Delta\Omega(\overrightarrow{\sigma})$ is the fraction of a solid angle associated to the direction $\overrightarrow{\sigma}$.

This discrete equation is a large linear system with unknowns $B(x)$ but the $I(x, \overrightarrow{\sigma})$ depends on visibility and occlusions.

The way to obtain a discrete set of voxels from a continuous surface is described in [Mal94] for an implicit surface, and in [Mal02] for a mesh.

In [CM06], a solution of the linear system is obtained with optimal complexity. We use a slightly improved version of this method, by using bucket (or radix) sort (complexity $O(n)$) instead of quicksort. Thus final complexity is $O(N \times I \times D)$, where N is the number of voxels of the discrete surface, D is the number of directions used for sampling the sphere, and I is the number of iteration, or number of successive reflections used for indirect lighting. We emphasize that **the complexity is linear (with a small coefficient) with respect to the total number of rays traced**. In [PM08], an integer only discrete ray shooting is combined with the linear method for direct lighting of voxels to increase directions sampling from light sources.

The method has a potential as universal method for solving the general light transport problem with *BRDF*, but up to now only the diffuse case is implemented satisfactorily. Up until now, there was no satisfactory method for rendering the results. We solve this problem in this paper.

4. Optimal Complexity for Visibility

To solve Equation (1), a converging iterative method similar to the one used with classical patch-based radiosity can be used: it usually relies on Jacobi or Gauss-Seidel relaxation. If we consider the linear equation under its form $B = E + M.B$, where B is a vector of elements and M a matrix of factors, some properties of M ensures the sequence $B_{n+1} = E + M.B_n$ converges toward a limit, which is a solution of the discrete linear system. Roughly speaking, this is a transcription of light gathering, each iteration going a step further in light re-emission. Convergence is expected since light is progressively absorbed. Technically, each iteration consists in propagating packets of energy between mutually visible voxels.

Given a direction vector $(a,b,c) \in \mathbb{Z}^3$ with $a \geq b \geq c$, a notion of a $3D$ line has been proposed [DR95], as the set of points $(x,y,z) \in \mathbb{Z}^3$ such that

$$\mu \leq cx - az < \mu + \omega \text{ and } \mu' \leq bx - ay < \mu' + \omega'$$

where $\mu, \mu', \omega, \omega'$ are integers. Other cases can be deduced by symmetry. It is noteworthy that under this definition, a $3D$ discrete line represents the intersection between two discrete planes, each being the orthogonal extrusion of a $2D$ discrete line included in one of the coordinate planes. The connectivity is related to the thickness ω and ω' of these $2D$ lines. If the two $2D$ projections are naïve (resp. standard), the $3D$ line is called naïve (resp. standard).

Proposition 1 Let us denote by $\mathbb{Z}^3_*$ the set $\mathbb{Z}^3 \setminus \{(0,0,0)\}$. Given an integer vector $\overrightarrow{v} \in \mathbb{Z}^3_*$, the set $\mathbb{Z}^3$ can be partitioned into $3D$ discrete lines, whose direction vector is $\overrightarrow{v}$ (see Figure 3). Moreover, given a voxel $x \in \mathbb{Z}^3$, finding out which $3D$ discrete line in the partition the point x belongs to can be done in constant time.

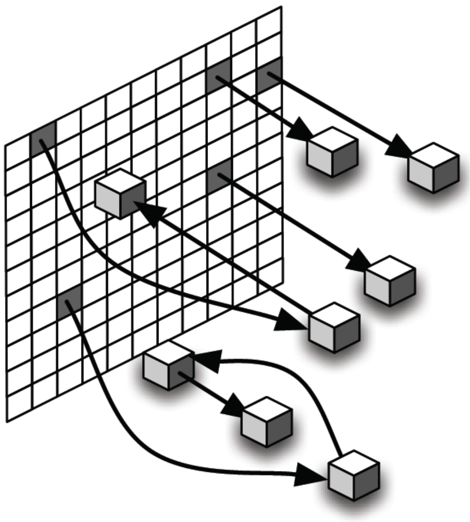
Now, solving our linear system given by Equation (1) by the Gauss-Seidel relaxation amounts to transferring energy from one voxel x to the first voxel y visible from x in a direction σ, for some finite set of sample directions σ. We can assume w.l.o.g that σ has integer coordinates. Now, if we consider some fixed direction σ and a partition of the voxel space $\mathbb{Z}^3$ into discrete lines parallel to σ, then the voxels of the discretized surface (say mesh or implicit surface) that lie on the same discrete line can be arranged in an ordered list. In this ordered list, the first visible voxel is the next voxel in the list (see Figure 2). So, once the lists are sorted, we can propagate the energy in linear time $O(N)$.

Now, the idea is that by going over the set of all surface voxels in a lexicographic order (lexicographic orders are precomputed by radix sort), we can build all the lists in linear time $O(N)$ (see Figure 3). We do this for each of the D sample directions and for each of the I Gauss-Seidel iteration, and we get a numerical solution of Equation (1) in time $O(N \times I \times D)$.

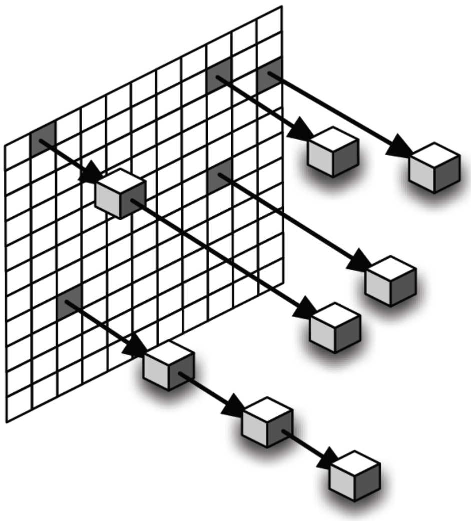
5. Display Method Using RayTracing

To compute the final image we use raytracing and use the voxels as a light sources (or $\mathbb{VPL}\sim$) in the way somewhat inspired by instant radiosity method [Kel97]. Of course, the principle must be substantially modified to be adapted to voxels. The intensity of the voxels is proportional to their radiosity as obtained in the output of the method of [CM06]. However, to take into account the nature of radiosity (power per unit of steradian per unit of area) we select the voxels *randomly* according to a probability proportional to their *area*, as defined below, and multiplied by a solid angle.

Each connected component χ of the continuous surface (e.g mesh) separates the space into two connected components, C and $\overline{C}$ (Jordan theorem), where $\overline{C}$ is the component which contains the viewpoint. We consider O the set of all the voxels $(i,j,k) \in Z^3$ which are contained in C (i.e. voxels inside an object or outside the scene). The discretization of the surface is the boundary F of O, which is the set of all



(a) Unsorted lists of voxels



(b) Sorted lists of voxels

Figure 2: *Lists of voxels obtained by intersection with discrete lines*

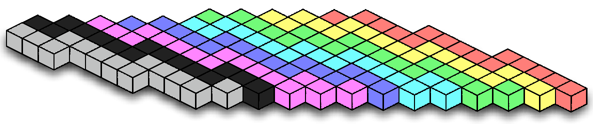


Figure 3: *The lexicographic on voxels coincides with linear order on rays*

voxels in O which are $6-$adjacent to (i.e. have a common face with) a voxel in the complement $\overline{O}$ of O.

For a voxel $x \in F$, the element of area $\Delta A(x)$ is the sum, for all $6-$neighbors y of x in $\overline{O}$, of the dot product $\overrightarrow{xy}.\overrightarrow{N}$, where $\overrightarrow{N}$ is the normal vector at the point x.

$$\Delta A(x) = \sum_{y \in N_6(x) \cap \overline{O}} \overrightarrow{xy}.\overrightarrow{N}$$

So, we make a raytracing phase by tracing rays from the viewpoint through the pixels. For each pixel, we compute the ray-object intersection point I that we must shade. In order to shade the point I, we use the sample voxels y, randomly selected with probability proportional to $\Delta A(y)$, as point light sources with intensity, or *energy contribution*:

$$C_y(I) = B(y)\,\frac{\cos\theta(I,y)\cos\theta(y,I)}{||I-y||^2}\,V(I,y)$$

In fact, we select two random samples sets of voxels: one for light sources and one for non light source voxels. The reason is the light sources (non-zero emittance) emit much more power and must be sampled more finely. We denote by $\mathcal{L}$ the set of all voxels with non-zero emittance (light source voxels) and $\mathcal{NL}$ the set of all non light-source voxels. We select a set $V_d \subset \mathcal{L}$ of light source sample voxels. and a set $V_i \subset \mathcal{NL}$ of non light source sample voxels. We denote $n_d = |V_d|$ the number of light source sample voxels and $n_i = |V_i|$ the number of non light source sample voxels, The total power at the pixel (or equivalently at I) is computed as :

$$\frac{1}{2\pi}\left(\frac{TAD}{n_d}.\sum_{y \in V_d} C_y(I) + \frac{TAI}{n_i}.\sum_{y \in V_i} C_y(I)\right)$$

where:

$$TAD = \sum_{x \in \mathcal{L}} \Delta A(y)$$

$$TAI = \sum_{x \in \mathcal{NL}} \Delta A(y)$$

are the total areas of light source surfaces $\mathcal{L}$ and non light source surfaces $\mathcal{NL}$.

We can add some specular effects at the point I using a local illumination model. We implemented successfully the *BRDF* model of ([AS00]).

6. Implementation

We use progressive raytracing that allows the user to get approximate results after a few seconds. For each frame, part of all the Virtual Point Lights (*VPL*) is taken into account. Thus at each instant the preview image is an approximation of the energy of the final solution. The output converges to the final solution and computation can be stopped by the user or automatically (by using a difference threshold between successive frames). Since we use the same set of *VPLs* for all the pixels (and possibly for all viewpoints in the same static scene), no noise is noticeable (the only problem is hard shadow edges with disappear after a sufficient number of *VPLs* is taken into account).

To create a proper *VPL* order, we select the voxels randomly according to a probability proportional to their *area*. To do this, we place all the voxels in a table, compute a random number between 0 to the sum of the areas ΔA, and use a binary search to select the right voxel. Conformly to theory of statistics, we do not reject a voxel after selection if selected more then once.

In the raytracing step, at each frame, for each pixel we compute the contribution of the next 16 of the *VPLs* to the final picture.

We add features like anti-aliasing and depth of field without additional cost, by slightly modifying the ray's position and/or direction. For both effects we use stratified sampling computed once for each frame and used by each pixel.

7. Results

Tests were done on an *Intel Core 2 Duo 6300* (1.86*GHz*). Only one core was used. Our computer has 2*GB* of memory (but generally less then 1*GB* was used). We used a completely home-made $CPU-$only raytracer, using $KD-$trees with a simplified surface-area heuristics. Our implementation of raytracing has many shortcomings, and so has our implementation of our voxel method. In particular, both are $CPU-$only and use no threads. Statistics about test scenes are shown in Table 1 and Table 2. Detailed number of rays per second are shown in Table 4.

Table 1: *Test scenes statistics (1)*

Scene	Voxels	LPL	SR	LR	Size	TT
Bunny1	146355	5	35	22	1024x768	558
Bunny30	146355	5	35	22	1024x768	2486
Bunny116	146355	5	35	22	1024x768	7742
Sponza	236754	7	35	26	1024x768	29185
Labirynth	11803	9	35	22	512x512	207
Emission	36062	4	8	22	512x512	112
Shadow	26547	4	22	18	512x512	147
Geometric	21074	4	28	18	512x512	208

These results inspire us two kinds of considerations.

1) With comparable implementations, the number of rays per second is by far greater for discrete rays than for continuous rays. This shows that a $GPU-CPU$ implementation of the linear discrete method will, beyond reasonable doubt, outrun $KD-$tree based ray-tracing. Moreover, the discrete method's performance is less sensitive to surfaces' complexity.

Table 2: *Statistics for different kinds of rays (2)*

Scene	ST	LT	RTT	SRays	LRays	CRays
Bunny1	93	404	52	70.5e6	807.6e6	9.45e6
Bunny30	93	404	1980	70.5e6	807.6e6	297e6
Bunny116	93	404	7236	70.5e6	807.6e6	1 133e6
Sponza	274	960	27755	211.4e6	1 181e6	2 030e6
Labirynth	12	68	124	22e6	183e6	48.5e6
Emission	5	50	56	3.7e6	284e6	8.7e6
Shadow	4	23	119	3.2e6	141e6	29.3e6
Geometric	11	20	176	8.1e6	109e6	43.5e6

Table 3: *Detailed column names for Tables 1 and 2*

Name	Description
Scene	Scene name
Voxels	Number of voxels
LPL	Light path length
	(number of successive reflexions $+1$)
SR	Shooting discrete sphere radius
LR.	Linear method discrete sphere radius
Size	Width and height in pixels of the output image
TT	Total runtime (in seconds)
ST	Voxel shooting time (in seconds)
LT	Linear method propagation time (in seconds)
RTT.	Ray-tracing display time (in seconds)
CRays	number of continous rays (raytracing)
SRays	number of discrete rays (shooting)
LRays	number of discrete rays (linear method)
DSR	Discrete shooting rays
DLR	Discrete linear method rays
CR	Continuous rays (classical ray-object intersection)

2) The discrete method combined with display by classical ray-tracing provides both accurate global illumination and good rendering. When coupled with a good $GPU-CPU$ implementation of ray-tracing, it will be a very competitive global illumination method.

Conclusion

Our experimental results show that Voxel Based Global Illumination can be competitive for exact simulation and has a potential in the future for real-time animation of static scenes. It has a better complexity than any other known accurate method for indirect lighting. Planned future work include plugging the method into some real-time *GPU* raytracer, developing effective techniques for non-static scenes and including non-lambertian model, including caustics in the simulation.

References

[AS00] Michael Ashikhmin and Peter Shirley. An anisotropic Phong BRDF model. *Journal of Graphics Tools: JGT*, 5(2):25–32, 2000.

Table 4: *Rays per second*

Scene	DSR	DLR	CR	CR/DSR	CR/DLR
Bunny30	758 064	1 999 010	150 105	19.8%	7.5%
Sponza	771 533	1 230 208	73 129	9.5%	5.9%
Labirynth	1 833 333	2 691 176	391 373	21.2%	14.5%
Emission	670 556	5 636 372	154 896	23%	2.7%
Shadow	703 332	5 886 649	246 006	35%	4.2%
Geometric	740 149	5 453 033	247 269	33.4%	4.5%

[CM06] Pierre Y. Chatelier and Remy Malgouyres. A low-complexity discrete radiosity method. *Computers & Graphics*, 30:37–45, February 2006.

[DR95] Isabelle Debled-Rennesson. *Étude et reconnaissance des droites et plans discrets*. PhD thesis, Université Louis Pasteur, Strasbourg, 1995. PhD thesis.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien.

[Jen97] Henrik Wann Jensen. Rendering caustics on non-Lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997.

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the ACM SIGGRAPH Conference (SIGGRAPH-97)*, pages 49–56, New York, August 3–8 1997. ACM Press.

[Mal94] R. Malgouyres. *Une dfinition des surfaces de* Z^3. PhD thesis, Universit Clermont 1, 1994.

[Mal02] Rémy Malgouyres. A discrete radiosity method. In Achille J.-P. Braquelaire, Jacques-Olivier Lachaud, and Anne Vialard, editors, *DGCI*, volume 2301 of *Lecture Notes in Computer Science*, pages 428–438. Springer, 2002.

[PM08] Łukasz Piwowar and Rémy Malgouyres. Combining shooting and gathering for voxel global illumination. Technical report, Universit Clermont 1, 2008. Submitted for publication.

[SJ00] B. Smits and H. Jensen. Global illumination test scenes. Technical report, University of Utah, 2000. Technical report.

[SP94] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, San Francisco, 1994.

[ZFM06] Rita Zrour, Fabien Feschet, and Rémy Malgouyres. Parallelization of a discrete radiosity method using scene division. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (2)*, volume 4276 of *Lecture Notes in Computer Science*, pages 1213–1222. Springer, 2006.