A New Mathematical Development for Radiosity Animation with Galerkin

Didier Arquès, Venceslas Biri, Sylvain Michelin

A New Mathematical Development for Radiosity Animation with Galerkin

Arquès Didier, Biri Venceslas, Michelin Sylvain
University of Marne-La-Vallée
Institut Gaspard Monge
5, Boulevard Descartes, F-77454 Champs sur Marne, France
{arques,biri,michelin}@univ-mlv.fr

Abstract

Combining animation and global illumination constitutes, at present, a true challenge in computer graphics, especially when light sources move in a complex scene because the entire illumination has to be recomputed. This paper introduces a new algorithm, based on the Galerkin method, which can efficiently manage any moving surface -even light source- to compute animation sequences. For each new frame of a sequence, we take into account the continuous property of the moves to determine the necessary energy differences between the previous global illumination solution and the new one. Based on a mathematical development of the form factor, this new approach leads to an efficient and simple algorithm, similar to the classical progressive refinement algorithm, and which computes animated sequence about three times faster.

1. Introduction and previous work

The use of realistic global illumination is becoming more and more widespread. But who has never seen, in the actual computer-animated movies, images which seem to be too artificial or shiny, due to inadequate lighting ambiance ? This is a consequence of the use of direct illumination algorithms. So there is a need for algorithms that can efficiently combine global illumination and dynamic objects because it really ”brings life” into a scene. But it remains one of the most difficult challenges in computer graphics because in a complex scene, dynamic objects (and especially moving light sources) cast new shadows, modify direct illumination and, indirectly, the whole energy relationships between objects.

For static environment, radiosity algorithm constitutes one of the most used means to solve the Kajiya's equation [17], basic equation of any global illumination model. Initially in the traditional radiosity algorithm [8, 13, 22], the radiosity function is projected onto some discrete finite bases. A first animation algorithm, the ”back buffer” [2] method, comes directly from this approach and allows to manage any known move with a static camera. Progressive methods [7, 28] lead to new interactive algorithms [5, 12] that propagate light modifications by shooting positive and negative luminous energy. For each move, two steps have to be carried out, one for withdrawing the object and one for adding it, what is unsuitable for moving light source. A more involved data structure has appeared [21] for this progressive refinement radiosity.

Probabilistic methods [18] have also been presented to manage very complex scenes and lead to other animation algorithms [3]. In the same time, the complexity of the function bases used to represent the illumination function in radiosity has been reduced by hierarchical and adaptive methods [4, 15, 19, 20]. Finally other function bases [1, 14, 23, 27, 29, 30] have been found to represent the illumination function but few animation methods based on it have been presented until now. Some animation algorithms [10, 11] have tried to use the benefits offered by adaptive bases, by identifying the links affected by an object displacement.

We present here a new radiosity algorithm, which manages the case of surfaces or light sources in translation. This method takes advantage of the continuity properties of these displacements to compute efficiently all illumination relationships between objects. We obtain a new algorithm similar to classical progressive refinement algorithm with new form factors, which allows to obtain quickly the new global illumination. As any radiosity algorithm, it can also be used as a first pass of a classical rendering system that can add specular and reflection effects.

The next section of this paper briefly returns on the Galerkin algorithm. In the third section, we are presenting the mathematical development and approximations we have used to treat a dynamic scene, while the fourth section shows results obtained by this method.

2. Theoretical Background : the Galerkin Algorithm

Galerkin radiosity [30] is an alternative radiosity formulation that generalizes the classical radiosity approach. This model avoids the last rendering interpolation and allows representing the scene with a restricted number of parametric surfaces. It avoids any discretisation, allows to store every kernel coefficients and separates shadow computation from illumination determination. Unlike the classical radiosity approach, as shown in figure 1, each function used (the radiosity, the exitance and the reflectivity) is projected onto a base of *N'* orthogonal non-constant functions $\{\Gamma_k(s,t) = 1..N'\}$ defined over each entire surface. For example, Zatz [30] uses a base of Legendre polynomials. Then, the Galerkin method consists in computing, for each surface, the radiosity coefficients associated to each function of the base.

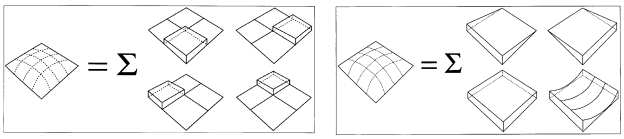

Figure 1. Decomposition on a classical discrete constant function basis and a higher order function basis

More precisely, let us consider a scene defined by *N* parametric surfaces. For a surface of index *i* and parameters *(s,t)*, the radiosity equation is :

$$B_i(s,t) = E_i(s,t) + \sum_{j=1}^{N} \iint K_{ij}(s,t,u,v) B_j(u,v)\, du\, dv \quad (1)$$

where E_i is the exitance and where the kernel function *K* is the product of the elementary form factor *F*, the reflectivity ρ, the area *A* and a visibility term *VIS* :

$$K_{ij}(s,t,u,v) = \rho_i(s,t)\ F_{i\to j}(s,t,u,v)\ VIS_{ij}(s,t,u,v)\ A_j(s,t) \quad (2)$$

Each function used, respectively the radiosity, the exitance and the reflectivity, is then projected onto a base of *N'* orthogonal functions $\{\Gamma_k(s,t), k = 1..N'\}$. For instance, radiosity becomes :

$$B(s,t) \approx \sum_{k=1}^{N'} b^k \Gamma_k(s,t) \quad (3)$$

where b_k is the coefficient associated to the k^{th} function of the base.

For each surface i, b_i^k coefficients are obtained by substituting (3), and respectively expressions of exitance and reflectivity, in equation (1) and by using the classical inner product between the radiosity $B_i(s,t)$ and the k^{th} function of the base. We obtain :

$$b_i^k = e_i^k + \sum_{j=1}^{N} \sum_{l=1}^{N'} b_j^l \langle \iint K_{ij}(s,t,u,v) \Gamma_l(u,v) | \Gamma_k(s,t) \rangle \quad (4)$$

and finally :

$$b_i^k = e_i^k + \sum_{j,l} b_j^l\ K_{ij}^{kl} \quad (5)$$

with

$$K_{ij}^{kl} = \langle \iint K_{ij}(s,t,u,v) \Gamma_l(u,v) | \Gamma_k(s,t) \rangle \quad (6)$$

This coefficient represents a kind of generalised form factor expressing the energy exchanged between the k^{th} function associated with surface i and the l^{th} function associated with surface j. Methods for computing the K_{ij}^{kl} terms use either traditional rules of quadratic integration [30], or Monte Carlo's techniques [18]. Equation (5), where the unknown are the radiosity coefficients b_i^k for each surface, can be solved by using indifferently a traditional direct numerical method, or via any progressive refinement techniques [7]. Indeed, since the surface indices i,j and function indices k,l are independent of each other, (5) is still a linear equation. Moreover, in the Galerkin method, shadow is moving out of the light calculation (visibility *VIS* is equal to 1). Zatz uses shadow masks but other shadow algorithms [6, 9, 16, 24, 25, 26] could be used.

3. Our approach

In this section, we present a new algorithm for producing a large animation sequence with global illumination determination. Our main goal is to compute at time $T + \Delta T$, the new illumination of a scene knowing the whole radiosity of the previous frame, at time T. It leads to a new progressive algorithm that computes only radiosity variations with new approximated form factors. It can manage any kind of surface -even light source- in any complex translation.

In the first time, we analyze the differences between two successive frames. Then we focus on the determination of the new form factor and the mean to obtain them quickly. Then we discus some discontinuity situations which deserve special treatments. Finally, we set our algorithm used to compute animation sequence.

3.1. Radiosity modifications between two frames

To simplify, let us consider the contribution of one moving surface j to surface i as shown in figure 2. Time dependent values are representing by adding a quote at time $T + \Delta T$.

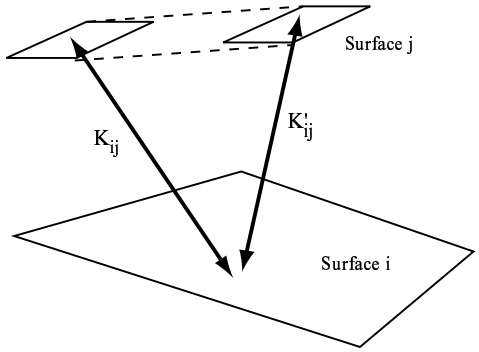

Figure 2. Study case

The fundamental idea is to find a mathematical relation between $b_i'^k$ and b_i^k. Expressing equation (1) for the two surfaces i and j, the radiosity difference is :

$$B'_i(s,t) - B_i(s,t) = E'_i(s,t) - E_i(s,t) + \iint_{u,v} [K'_{ij} B'_j - K_{ij} B_j](s,t,u,v)\, du\, dv \quad (7)$$

Using notation $K'_{ij} = K_{ij} + \Delta K_{ij}$, $B'_i = B_i + \Delta B_i$, $E'_i = E_i + \Delta E_i$ equation (7) could be reformulated :

$$\Delta B_i = \Delta E_i + \iint \Delta K_{ij}.B_j + \Delta B_j.(K_{ij} + \Delta K_{ij}) \quad (8)$$

If we develop the radiosity B_j by substituting (3) in equation (8) and projecting the result onto the k^{th} base function, a relation is obtained :

$$\Delta b_i^k = \Delta e_i^k + \sum_{l=1}^{N'} b_j^l \Delta K_{ij}^{kl} + \sum_{l=1}^{N'} \Delta b_j^l.(K_{ij}^{kl} + \Delta K_{ij}^{kl}) \quad (9)$$

with

$$\Delta K_{ij}^{kl} = \left\langle \iint_{u,v} (K'_{ij} - K_{ij})(s,t,u,v) \Gamma_l(u,v) \mid \Gamma_k(s,t) \right\rangle \quad (10)$$

We should notice that equation (9) can be reversed to determine also the modification in illumination of the moving surface j - created by is own move - from static surface i. We just have to consider that surface i have a virtual relative move opposite to the original move of surface j and compute the coefficient ΔK_{ji}.

3.2. Calculation of coefficients ΔK

Henceforth, the main problem is to determine for each frame the new coefficients ΔK. If we consider that the surface j follows a translation movement in a direction $\vec{p_0}$ as shown in figure 3, the idea is to obtain, with a limited development in p of the expression ΔK, a polynomial expression of ΔK like :

$$\Delta K = \sum_{n \geq 1} p^n \varphi_n$$

where φ_n depend only on $\vec{p_0}$.

If, from an initial reference position, each coefficient φ is computed, then for each displacement p of the surface j in the direction $\vec{p_0}$, ΔK will be efficiently computed.

We start with the expression of K in equation (2) with terms :

$$F_{i \to j}(s,t,u,v) = -\frac{\vec{n_d}.\vec{r'}\,\vec{n_s}.\vec{r'}}{\pi \vec{r'}^4} \text{ and } A_j(u,v) = \left\| \frac{\delta \vec{N}}{\delta u} \wedge \frac{\delta \vec{N}}{\delta v} \right\|$$

So we have :

$$K'_{ij} = -\frac{\rho_i}{\pi} \frac{\vec{n_d}.\vec{r'}\vec{n'_s}.\vec{r'}}{\vec{r'}^4} \left\| \frac{\delta \vec{N}}{\delta u} \wedge \frac{\delta \vec{N}}{\delta v} \right\| \quad (11)$$

Let consider the expression of r'^4 :

$$\vec{r'}^4 = (\vec{r} + \vec{p})^4 = \vec{r}^4.\left(1 + 2\frac{\vec{p}.\vec{r}}{r^2} + \frac{\vec{p}^2}{r^2}\right)^2 \quad (12)$$

We denote $\alpha = \vec{p_0}.\vec{r}$ and $\beta = \vec{p_0}^2$. The limited development of the inverse of equation (12) gives :

$$\frac{1}{r^4} \sum_{n \geq 0} (-1)^n.(n+1).\left(2\frac{\alpha}{r^2}p + \frac{\beta}{r^2}p^2\right)^n$$

and is defined only if :

$$r > (1 + \sqrt{2}).p \quad (13)$$

Using the following notation :

$$\begin{cases} a = (\vec{n_d}.\vec{r}).(\vec{n_s}.\vec{r}) \\ b = (\vec{n_d}.\vec{p_0}).(\vec{n_s}.\vec{r}) + (\vec{n_d}.\vec{r}).(\vec{n_s}.\vec{p_0}) \\ c = (\vec{n_d}.\vec{p_0}).(\vec{n_s}.\vec{p_0}) \end{cases}$$

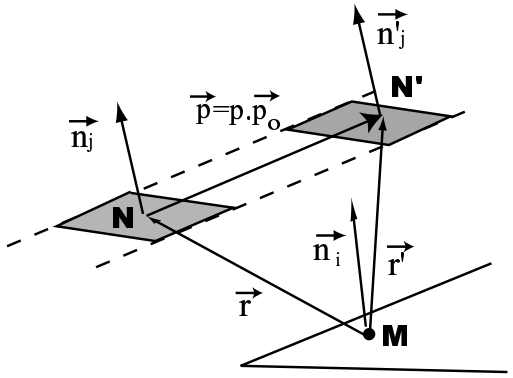

Figure 3. Details of a moving surface and a static surface

we have

$$K'_{ij} = -\frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.\left(a+b.p+c.p^2\right). \sum_{n\geq 0}(-1)^n.(n+1).\left(2\frac{\alpha}{r^2}p+\frac{\beta}{r^2}p^2\right)^n \quad (14)$$

or

$$K'_{ij} = K_{ij} - \frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.(b.p+c.p^2) - \frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.(a+b.p+c.p^2). \sum_{n\geq 1}(-1)^n.(n+1).\left(2\frac{\alpha}{r^2}p+\frac{\beta}{r^2}p^2\right)^n$$

Finally p is extracted from the last term of the preceeding equation :

$$\begin{aligned}
&\sum_{n\geq 1}\frac{-1^n.(n+1)}{r^{2n}}\sum_{k=0}^{n}\binom{n}{k}(2\alpha p)^{n-k}\beta^k p^{2k}\\
&=\sum_{n\geq 1}\frac{-1^n.(n+1)}{r^{2n}}\sum_{k=0}^{n}\binom{n}{k}(2\alpha)^{n-k}\beta^k p^{k+n}\\
&=\sum_{n\geq 1}\frac{-1^n.(n+1)}{r^{2n}}\sum_{q=n}^{2n}\binom{n}{q-n}(2\alpha)^{2n-q}\beta^{q-n}p^q\\
&\quad avec\ \ q=n+k\\
&=\sum_{n\geq 1}\sum_{q=n}^{2n}\frac{-1^n.(n+1)}{r^{2n}}\binom{n}{q-n}(2\alpha)^{2n-q}\beta^{q-n}p^q\\
&=\sum_{q\geq 1}p^q\sum_{n=\lceil\frac{q}{2}\rceil}^{q}\frac{-1^n.(n+1)}{r^{2n}}\binom{n}{q-n}(2\alpha)^{2n-q}\beta^{q-n}\\
&=\sum_{q\geq 1}p^q\ \xi_q
\end{aligned}$$

and

$$\begin{aligned}
&(a+b.p+c.p^2).\sum_{q\geq 1}p^q\xi_q\\
&=a\sum_{q\geq 1}p^q\xi_q+b\sum_{q\geq 1}p^{q+1}\xi_q+c\sum_{q\geq 1}p^{q+2}\xi_q\\
&=a\,\xi_1\,p+a\,\xi_2\,p^2+b\,\xi_1\,p^2+\sum_{q\geq 3}p^q(a\xi_q+b\xi_{q-1}+c\xi_{q-2})
\end{aligned}$$

We have then

$$K'_{ij}-K_{ij}=\sum_{n\geq 1}p^n\varphi_n^{ij}(s,t,u,v,\vec{p_0}) \quad (15)$$

with

$$\begin{cases}
\varphi_1^{ij} = -\frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.(a\xi_1+b)\\
\varphi_2^{ij} = -\frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.(a\xi_2+b\xi_1+c)\\
\forall n\geq 3, \varphi_n^{ij} = -\frac{\rho_i}{\pi r^4}.\left\|\frac{\delta\vec{N}}{\delta u}\wedge\frac{\delta\vec{N}}{\delta v}\right\|.(a\xi_n+b\xi_{n-1}+c\xi_{n-2})\\
\forall n, \xi_n = \sum_{q=\lceil\frac{n}{2}\rceil}^{n}(q+1).\binom{q}{n-q}.\left(\frac{-4\alpha^2}{\beta r^2}\right)^q.\left(\frac{\beta}{2\alpha}\right)^n
\end{cases}$$

And finally,

$$\Delta K_{ij}^{kl}=\sum_{n\geq 1}p^n\left\langle\iint_{u,v}\varphi_n^{ij}(s,t,u,v,\vec{p_0})\Gamma_l(u,v)\mid\Gamma_k(s,t)\right\rangle \quad (16)$$

Equation (16) allows to compute efficiently coefficients ΔK. Indeed, coefficients φ are recursively defined and depend only on the direction $\vec{p_0}$, on the degree of approximation n, and on both surfaces i,j and their function k,l. So coefficients φ could be computed in advance for any elementary directions, like axes $\vec{x}$, $\vec{y}$ and $\vec{z}$, and used for all move p in these directions.

3.3. Condition of validation and discontinuity

The equation (16) is usable only if condition (13) is satisfied and if discontinuities have not occurred because continuity is an implicit condition of all previous calculations.

Discontinuities occur when a receiving surface, or a part of it, enters or goes out of the visibility of the moving surface. For discontinuities, we have to check if receiving surface enters or goes out of the positive half space defined by the emitter, as shown in figure 4. Simple geometrical computations, like BSP Tree [6], can determine such particular situations.

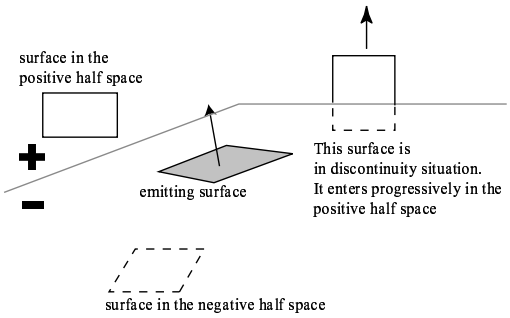

Figure 4. Example of positive and negative half space of an emitting surface

When discontinuity occurs, we have to compute the new coefficients K' explicitely using equation (6). At this time, we get the correct value of K and errors, involved by multiple addition of ΔK, disappear.

The condition (13) expresses that, for a particular elementary direction $\overrightarrow{p_0}$, our approximation is defined near the reference point where we compute the limited development and the coefficients φ. r represents the distance between moving surface and static surface at this reference position and p the norm of the path of the moving surface from it. So these two values are stored for each direction and each receiving surface, and updated to control condition (13).

When this condition is no more satisfied in a particular direction, we have to compute explicitely K' and also the coefficients φ. r and p are also updated (always for that direction). Once computed, the new coefficients φ remain valid until the new condition (13) is not satisfied i.e. moving surface moves too far in that direction.

3.4. Illumination of the scene for a new frame

The following system of equation allows to compute incrementally the global illumination solution of a scene :

$$\Delta b_i^k = \Delta e_i^k + \sum_{l=1}^{N'} b_j^l \Delta K_{ij}^{kl} + \sum_{l=1}^{N'} \Delta b_j^l . (K_{ij}^{kl} + \Delta K_{ij}^{kl})$$

$$with$$

$$\Delta K_{ij}^{kl} = \left\langle \iint_{u,v} (K'_{ij} - K_{ij})(s,t,u,v)\Gamma_l(u,v) \mid \Gamma_k(s,t) \right\rangle$$

This system is equivalent to the galerkin radiosity equation (5) and can be seen as a progressive equation with the unknown variables Δb. There are only two differences :

- The emission term has been replaced by an eventual change in exitance Δe_i^k, plus a sum $b_j^l \Delta K_{ij}^{kl}$ that represents the illumination difference involved by the move.
- The unknown variables have changed from b_i to Δb_i.

Every complex path, like splines, could be decompose in a sequence of little moves along elementary directions (the three axes for example). Then for each new image, we determine directions and step of the moving surfaces (we don't have to know the movement in advance). Depending on condition (13) and discontinuity, we use equations (16) or (6) to compute the two first terms of equation (9). Computing these terms initialises the progressive refinement algorithm, that we use to propagate the light in the scene with the new form factors. When all the light have been propagated, we have the illumination of the entire scene without shadow effects. To represent them, we use the algorithm of the Shadow Volume Binary SPace Tree (SVBSP Tree) [6]. We choose this algorithm for its simplicity and its efficiency. Indeed, hard shadows are obtained very quickly and the realism generated by the Galerkin method is enough to obtain visually correct results. Other shadow algorithms [9, 16, 25] could be used.

Aliasing appears in the animation sequence but few modifications allow to erase it. Aliasing is due to discretisation of the scene and of the radiosity solution (and especially shadows). Jittering avoids the first kind of aliasing and takes only one second (or less with graphic cards allowing accumulation buffer). The radiosity solution over a surface is represented with a regular grid which involve also aliasing. We can use a finer grid or jitter it. Memory cost will limit the first approach but will be quicker. Jittering the grid forces redrawing the scene four or eight times, so will take about 1 or 2 seconds per image.

4. Results

The main computation time in the galerkin algorithm, except for shadow mask, is due to the determination of pseudo form factor. Our algorithm avoids computing them for each frame. In fact we made a costly computation in the beginning to obtain coefficients φ but then for each image of the sequence we don't have to compute explicitly the new form factors. Moreover, between two positions, we only compute the difference of energy involved by the move. Indeed, we calculate what is really necessary since we use a progressive method for each position (normal progressive method) and also between each position (our temporal progressive method). Thus, we can manage complex dynamic scene easily and simply since the main part of these scenes will probably not be concerned by the energy exchanges involved by the move (like the bressenham algorithm). Notice also that any complex path could be represented if it can be decomposed in little elementary moves. And we doesn't have to know the move in advance.

We present here some results and images from animation sequences computed with a 500 MHz processor and a common PC graphic card. The first scene, illustrated in figure 5.a, is a simple scene with a blue surface passing in front of a light and a second "ceiling light" moving left and right in translation. The right wall is correctly lighted in blue when the blue occluder enters just in front of the light. The red wall in the left receives also more light when the ceiling light comes nearest. Table 1 shows computation times and benefits of our method. We shall point out that this scene constitutes a worst case because the blue panel is very close to the light (and so r is small) and it enters progressively in the light positive half space.

100 frames of figure 5.a	Computation of φ	Time / frame	Total time
Galerkin progressive method	/	4.02 s	402 s
Our method	18.59 s	1.41 s	160 s
Benefits	/	65 %	60 %

Table 1. Result for scene of figure 5.a

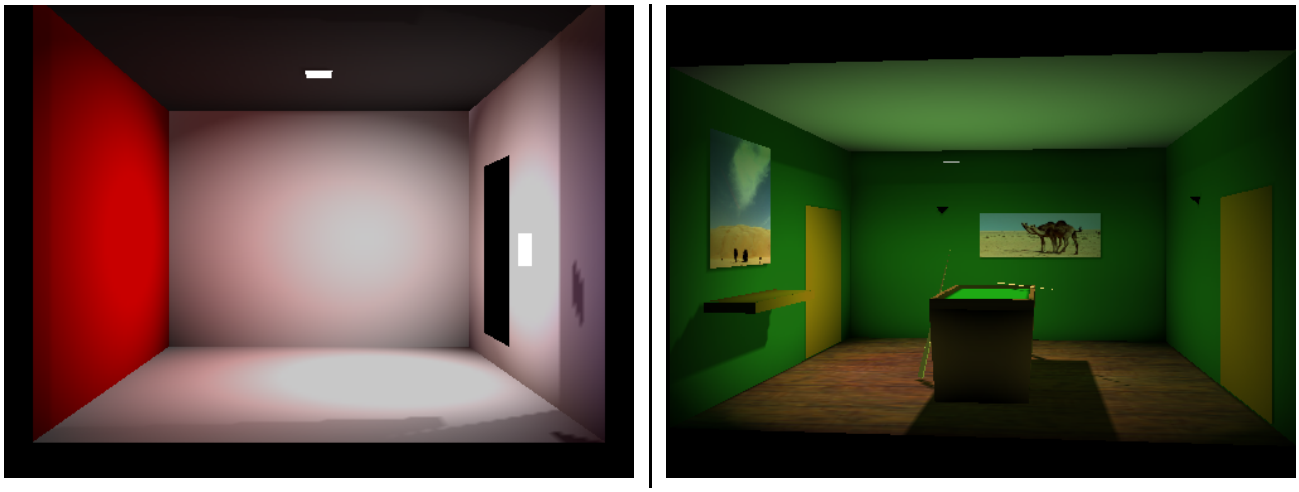

Figure 5. a. Simple Scene b. Complex scene : The billard

The second scene, shown in figure 5.b, is a complex scene with a moving light. We shows in table 2 computation times for that particular scene. Despite multiple discontinuities that occur, timesaving remains the same. We also present in the animation movies a more complicated scene (figure 6), a cathedral interior, with 140 surfaces representing about 110.000 polygons. Time saving is still equivalent to the previous scene. We should notice that we are able to represent any complex move in translation unlike what is showed in the animations.

160 frames of figure 5.b	Computation of φ	Time / frame without Stree	Time / frame	Total time
Galerkin method	/	2.93 s	2.03 s	469 s
Our method	18.59 s	1.13 s	0.24 s	185 s
Benefits (%)	/	61 %	88 %	60 %

Table 2. Result for scene of figure 5.b

5. Conclusion and perspective

In this paper, we have presented a new algorithm that allows to solve the case of any surface in translation -even light source- in the radiosity method. Timesaving is obtained by avoiding the computation in each frame of the sequence, for each moving object, the form factors associated with the mobile elements. Moreover, this algorithm makes only the necessary calculations what gives us a very efficient time for each new frame computation. Perspectives relate to shadows. We concentrated only on hard shadow but other algorithms could be used to represent soft shadows. As the Galerkin method does not require any discretisation of the scene, the storage cost of information of visibility can be managed. Another subject of development relates to the rotation effects of an emitting surface. The problem is more complicated since a slight rotational movement can involve major change in the illumination. Nevertheless, with appropriate limitations, for small oscillations, we can obtain a correct and useful approximation.

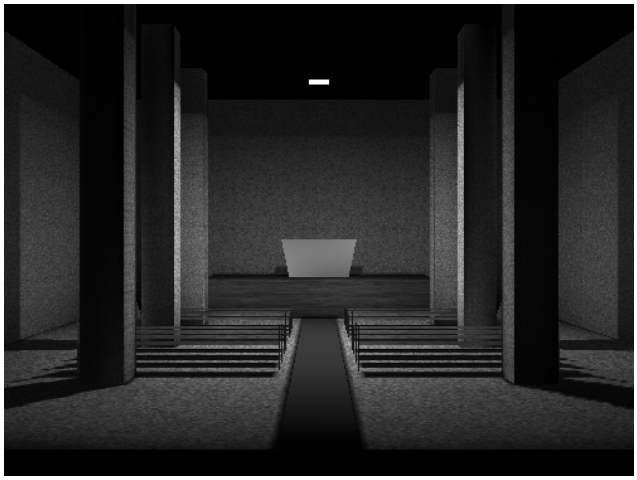

Figure 6. the cathedrale scene

References

[1] D. Arques, S. Michelin, and B. Piranda. Using a New Function Base with Local Disk Support. In *WSCG'2000 conference proceeding*, volume 2, pages 236–243, 2000.

[2] D. R. Baum, J. R. Wallace, M. F. Cohen, and D. P. Greenberg. The Back Buffer Algorithm : An Extension of the Radiosity Method to Dynamic Environments. In *Visual Computer*, volume 2(5), pages 298–308, 1986.

[3] G. Besuievsky and M. Sbert. The Multi-Frame Lighting Method : a Monte Carlo Based Solution for Radiosity in

Dynamic Environments. In *7th Eurographics Workshop on Rendering*, pages 186–195, June 1996.

[4] A. T. Campbell and D. S. Fussell. Adaptive Mesh Generation for Global Diffuse Illumination. In *Siggraph'90, Computer Graphics, Conference Proceeding*, volume 24(4), pages 155–164, Aug. 1990.

[5] S. Chen. Incremental Radiosity : an Extension of Progressive Radiosity to an Interactive Image Synhtesis System. In *Siggraph'90, Computer Graphics Conference Proceeding*, volume 24(4), pages 135–144, Aug. 1988.

[6] N. Chin and S. Feiner. Near Real Time Shadow Generation using BSP Trees. In *Siggraph'89, Computer Graphics*, volume 23(3), pages 99–106, 1989.

[7] M. Cohen, S. Chen, J. Wallace, and D. Greenberg. A Progressive Refinement Approach for Fast Radiosity Image Generation. In *Siggraph'88, Computer Graphics*, volume 22(4), pages 74–84, 1988.

[8] M. F. Cohen and D. P. Greenberg. The Hemi-Cube : A Radiosity Solution for Complex Environments. In *Siggraph'85, Computer Graphics*, volume 19(3), pages 31–40, 1985.

[9] G. Drettakis and E. Fiume. A Fast Algorithm for Area Light Source Using Backprojection. In *Siggraph'94, Computer Graphics, Annual Conference Series*, pages 223–230, July 1994.

[10] Y. Dupuy, F. Lavignotte, and M. Paulin. Visibilité et Radiosité Interactive. In *12eme journée de l'AFIG. AFIG'99 Conference Proceeding*, pages 247–257, Nov. 1999.

[11] D. Forsyth, C. Yang, and K. Teo. Efficient Radiosity in Dynamic Environments. In *Proceeding of 5th Eurographics on Rendering*, June 1994.

[12] D. W. George, F. X. Sillion, and D. P. Greenberg. Radiosity Redistribution for Dynamic Environment. In *IEEE Computer Graphics*, volume 10(4), pages 26–34, 1990.

[13] C. Goral, K. Torrance, D. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. In *Siggraph'84, Computer Graphics*, volume 18(3), pages 213–222, 1984.

[14] S. J. Gortler, P. Schroder, M. F. Cohen, and P. Hanrahan. Wavelet Radiosity. In *Siggraph'93, Computer Graphics Proceedings, Annual Conference Series*, volume 27(4), pages 221–230, Aug. 1993.

[15] P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25(4), pages 197–206, July 1991.

[16] P. Heckbert and M. Herf. Simulating Soft Shadows with Graphics Hardware. In *Technical report TR CMU-CS-97-104, Carnegie Mellon University*, Jan. 1997.

[17] T. Kajiya. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20(4), pages 143–150, Aug. 1986.

[18] A. Keller. Quasi Monte Carlo Radiosity. In *7th Eurographics Workshop on Rendering*, pages 102–111, June 1996.

[19] D. Lischinski, F. Tampieri, and D. P. Greenberg. Discontinuity Meshing for Accurate Radiosity. In *IEEE Computer Graphics*, volume 12(6), pages 25–39, Nov. 1992.

[20] D. Lischinski, F. Tampieri, and D. P. Greenberg. Combining Hierarchical Radiosity and Discontinuity Meshing. In *Siggraph'93, Computer Graphics Proceeding*, pages 199–208, Aug. 1993.

[21] S. Muller and F. Schoffel. Fast Radiosity Repropagation For Interactive Virtual Environments Using A Shadow-Form-Factor-List. In *5th Eurographics Workshop on Rendering*, 1994.

[22] T. Nishita and E. Nakamae. Continuous Tone Representation of Tree-Dimentional Object Taking Account of Shadows and Interreflection. In *Siggraph'85, Computer Graphics*, volume 19(3), pages 23–30, July 1985.

[23] S. N. Pattanaik and K. Bouatouch. Fast Wavelett Radiosity Method. In *Eurographics'94*, volume 13(3), pages 407–420, 1994.

[24] M. Segal, C. Korobin, R. V. Widenfelt, J. Foran, and P. Haeberli. Fast Shadows and Lighting Effects Using Texture Mapping. In *Siggraph'92, Computer Graphics Conference Proceeding*, volume 26(2), pages 249–252, July 1992.

[25] C. Soler and F. X. Sillion. Fast Calculation of Soft Shadow Textures Using Convolution. In *Siggraph'98, Computer Graphics, Annual Conference Series*, pages 321–332, 1998.

[26] J. A. Steward and S. Ghali. Fast Computation of Shadow Boundaries Using Spatial Coherence and Backprojections. In *Siggraph'94, Computer Graphics, Annual Conference Series*, pages 231–238, 1994.

[27] R. Troutman and N. L. Max. Radiosity Algorithms Using Higher Order Finite Element Methods. In *Siggraph'93, Computer Graphics, Annual Conference Series*, pages 209–212, 1993.

[28] J. Wallace, K. Elmquist, and E. Haines. A Ray Tracing Algorithm for Progressive Radiosity. In *Siggraph'89, Computer Graphics*, volume 23(3), pages 315–324, 1989.

[29] Y. Yizhou and P. Qunsheng. Multiresolution B-Spline Radiosity. In *Eurographics'95*, volume 14(3), pages 285–298, 1995.

[30] H. R. Zatz. Galerkin Radiosity: A Higher Order Solution Method for Global Illumination. In *Siggraph'93, Computer Graphics Proceedings, Annual Conference Series*, pages 213–220, 1993.