Dynamic Radiosity using Higher Order Functions Bases and Temporal Coherence

Biri Venceslas Michelin Sylvain Arquès Didier
University of Marne-La-Vallée
6 cours du Danube
F-77700 Serris, France
biri@univ-mlv.fr michelin@univ-mlv.fr arques@univ-mlv.fr

ABSTRACT

The computation of global illumination in a dynamic scene constitutes a real challenge in computer graphics. In radiosity algorithms, this problem is far from being easy, especially when light sources move in a complex scene. This subject becoming more and more widespread, many algorithms have been presented to solve the dynamic radiosity problem. Unfortunately, none uses intensive temporal coherence and few are efficient when dealing with a moving light source. This paper introduces a new algorithm that computes animations with any moving surfaces - even light sources. We take into account the temporal coherence between two frames to determine only the luminous energy differences between the previous global illumination solution and the new one. A mathematical development of the form factor for a translation or a rotation avoids unnecessary form factors computations. This new approach leads to an efficient and simple algorithm, similar to the classical progressive refinement method. Thus, it is able to compute the global illumination of animations at least twice faster than the classical approach.

Keywords

Dynamic Radiosity, Temporal Coherence

1. INTRODUCTION

According to the growing number of publications on the subject, the determination of global illumination in a dynamic scene appears to be the next challenge of computer graphics. Indeed, people want more and more realism in computer generated movies. For example, digital effects require an invisible contribution to the real video sequences and global illumination is the key to achieve such quality. Thus, there is a need for algorithms that can efficiently combine global illumination and dynamic objects, because it "brings life" into a scene. But this is a difficult task : in a complex scene, any moving surface - and especially light source - disturbs the whole illumination solution, modifying not only the luminous relationships between objects but also the occlusions.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

Radiosity algorithms constitute a very popular method to obtain a realistic illumination of a static scene. J. Kajiya [15] showed that radiosity is a particular approximation of the integral equation he called the rendering equation. In this equation, the radiosity of an object is defined as a function over its surfaces. Initially in the traditional radiosity algorithm [11, 5], this function is projected onto some piecewise constant functions over each surface. A first animation algorithm, the back buffer method [1], comes directly from this classical approach and allows to manage any known move with a static camera. Progressive methods [4, 21] are designed to solve the whole radiosity solution step by step. They lead to new interactive algorithms [3, 10] that propagate light modifications by shooting positive and negative luminous energy. In this approach, two steps have to be carried out for each single move, one for withdrawing the object and one for adding it, what is, of course, unsuitable for moving light sources. A more involved data structure has been proposed [18] for this progressive refinement strategy. The number of relationships between surfaces has been reduced, in the same time, by hierarchical and adaptive methods [13, 17]. This approach is used to handle dynamic scenes, with either a four-dimensional radiosity including time [6] or a hierarchy of energy

links [8, 9]. Probabilistic methods [16] were also presented to manage very complex scenes and they involve other animation algorithms [2]. Other function bases have finally been considered to represent the radiosity over a surface. Algorithms using these bases divide between the progressive approach [23] and the hierarchical one [12, 22]. Finally other researchs focus on light sources [7] and lead to algorithms able to handle moving light source [19]

We present here a new radiosity algorithm that is able to compute long animations using higher order functions bases in a dynamic scene, where surfaces - and even light sources - can move. We choose to use higher order functions bases to avoid discretisation problems but our method can apply to the classical progressive algorithm. Indeed, this method takes advantage of the continuity properties of these displacements and of their temporal coherence to avoid the computation of unnecessary form factors between two successive frames. Then, for each new frame, the light energy difference between any pair of surfaces is determined leading to a new algorithm, similar to the classical progressive refinement algorithm, where form factors are quickly updated. The use of temporal coherence allows us to obtain the new global illumination solution faster than the classical approach.
The next section of this paper briefly returns on the theoretical background of radiosity algorithms using higher order functions bases. In the third section, we present a progressive radiosity approach for dynamic scene and the mathematical developments we use to handle the temporal coherence of the moves. The fourth section set the resulting algorithm while the fifth discusses results obtained.

2. HIGHER ORDER RADIOSITY

Higher order radiosity algorithms are the algorithms that compute radiosity using a higher order functions base. The radiosity of an object is considered, in this approach, as a function defined on its surface instead of being constant over small patches. It allows the representation of the scene with a restricted number of parametric surfaces, which could avoid any discretisation.
If the N surfaces representing the scene are Lambertian diffuse, the rendering equation for a surface i can be written [12], using notations of figure 1, as :

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

where E_i and ρ_i are respectively the exitance and the reflectivity of the surface. The kernel function K is the product of the well-known form factor, the differential area A and a visibility term V :

$$K_{ij}(s,t,u,v) = -\frac{(\vec{n_i}.\vec{r})(\vec{n_j}.\vec{r})}{\pi \vec{r}^4} V_{ij}(s,t,u,v)\, A_j(u,v) \quad (2)$$

where

$$A_j(u,v) = \left\| \frac{\partial \vec{x_j}}{\partial u} \wedge \frac{\partial \vec{x_j}}{\partial v} \right\|$$

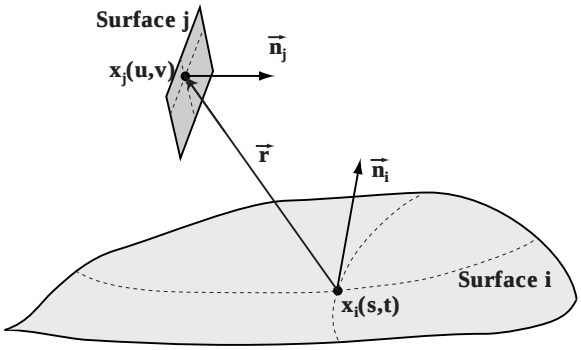

Figure 1. Notations for two parametric surfaces

Each function used in higher order algorithms - the radiosity and the reflectivity - is projected onto a base of N' orthonormal functions $\{\Gamma_k(s,t), k = 1..N'\}$ defined over each entire surface. Zatz [23] uses a base of Legendre polynomials when Gortler et al.[12] use wavelett bases. 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 scalar coefficient associated to the k^{th} function of the base. Now, the goal is to obtain the coefficients b^k of each surface to reconstruct their radiosity function. If we use an orthonormal base, it can be done with the inner product of two functions defined by :

$$\langle f|g \rangle = \int f(s)\, g(s)\, ds \quad (4)$$

We will consider the reflectivity constant over each object. Then, for each surface i, b_i^k coefficients will be obtained by substituting (3), and the similar expression of exitance, in equation (1). The result is then projected, using the classical inner product, on the k^{th} function of the base (in the following, we will omit dependance on parameters s, t, u and v) :

$$b_i^k = e_i^k + \rho_i \sum_{j=1}^{N} \sum_{l=1}^{N'} b_j^l \langle \iint K_{ij} \Gamma_l | \Gamma_k \rangle \quad (5)$$

and finally :

$$b_i^k = e_i^k + \rho_i \sum_{j,l} b_j^l\, K_{ij}^{kl} \quad (6)$$

with

$$K_{ij}^{kl} = \langle \iint K_{ij} \Gamma_l | \Gamma_k \rangle \quad (7)$$

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 can use either traditional rules of quadratic integration [23], Monte Carlo techniques [16], or closed form [20]. To avoid any singularity problem in integration, we choose to use both monte carlo and closed form techniques. Equation (6), where the unknowns are the radiosity coefficients b_i^k, can indifferently be solved by using a traditional direct numerical method, e.g. Gauss Seidel, or via any progressive refinement technique. Indeed, since the surface indices i,j and the function indices k,l are independent, equation (6) is still a linear equation.

Occlusions are finally treated in several ways. For example, H. Zatz chooses to compute them in shadow masks weighting the radiosity function. Gortler et al., using ray tracing as in [13], compute directly visibility coefficients which attenuate pure form factors.

3. OUR METHOD

We present, in this section, the mathematical foundation of progressive temporal radiosity inspired by George et al. [10]. The followed goal is to obtain the global illumination solution of a frame using intensively the illumination solution of the previous one. We choose to use higher order functions algorithms because they allow to represent the scene with a restricted number of parametric surfaces and avoid any discretisation. This makes possible to store every kernel coefficient and to separate shadow computation from illumination determination. Moreover, instead of computing new form factors, we will compute only their variations. An estimation of these variations can be found allowing instant determination of the new form factors for each frame and each surface. It leads to an efficient algorithm managing any kind of surface - even light source - in any complex move.

First, we analyse the differences of the radiosity solution between two successive frames. Then, we focus on the form factor variations, which depend on the object moves themselves. A method to obtain these variations quickly is proposed for translations and rotations.

3.1. Progressive radiosity between two frames

We are looking for radiosity variations between two successive frames at time T and $T+\Delta T$. Time dependent values at time $T+\Delta T$ will be marked by a quote ($'$). Writing equation (6) for the two frames gives :

$$\begin{cases} b_i^k = e_i^k + \rho_i \sum_{j,l} b_j^l \, K_{ij}^{kl} \\ b_i^{'k} = e_i^{'k} + \rho_i \sum_{j,l} b_j^{'l} \, K_{ij}^{'kl} \end{cases} \quad (8)$$

And using notations $K_{ij}^{'kl} = K_{ij}^{kl} + \Delta K_{ij}^{kl}$, $b_i^{'k} = b_i^k + \Delta b_i^k$ and $e_i^{'k} = e_i^k + \Delta e_i^k$, and subtracting previous equations gives :

$$\begin{aligned} \Delta b_i^k = \Delta e_i^k \; &+ \sum_{j,l} b_j^l \Delta K_{ij}^{kl} \\ &+ \sum_{j,l} \Delta b_j^l (K_{ij}^{kl} + \Delta K_{ij}^{kl}) \end{aligned} \quad (9)$$

with

$$\Delta K_{ij}^{kl} = \left\langle \iint_{u,v} (K'_{ij} - K_{ij}) \Gamma_l \mid \Gamma_k \right\rangle \quad (10)$$

Equation (9) is a generalisation for higher order algorithms of the radiosity redistribution equation defined by Georges et al. [10]. This equation is similar to the equation (6) but with two differences :

- Instead of b_i, the unknown variables are the Δb_i.
- The emission term has been replaced by a possible change of exitance Δe_i^k, plus a sum, $b_j^l \Delta K_{ij}^{kl}$, that is the radiosity variations induced by the move.

3.2. Determination of coefficients ΔK

Henceforth, the main problem is to determine for each frame the coefficients ΔK, which remain the only parameters we do not know. Our idea is to use temporal coherence between two frames to avoid the costly computations of new form factors concerning dynamic objects. Therefore, instead of computing these form factors, we try to approximate their variations using the surface moves. Our main contribution is to have obtained a precise expression of the form factors variations depending on the translation or on the rotation of the dynamic surfaces.

3.2.1 Computation of ΔK for a translation

Let us consider, to simplify, the contribution of the moving surface j to the surface i without occlusion ($V = 1$). If the surface j follows a translatory movement in a direction $\overrightarrow{p_0}$ as shown in figure 2, we can obtain a polynomial expression of the expression ΔK like :

$$\Delta K_{ij} = \sum_{n \geq 1} p^n \varphi_{ij}^n \quad (11)$$

where φ_n do not depend on p. At each frame, only p changes. So if we know the coefficients φ_n in a reference position, this formulation allows us to

compute the form factors variations in a constant time.

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

$$K'_{ij} = -\frac{(\vec{n_d}.\vec{r'})(\vec{n'_s}.\vec{r'})}{\pi \vec{r'}^4} A_j \qquad (12)$$

And let us consider the expression of r'^4 :

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

If we denote $\alpha = \vec{p_0}.\vec{r}$ and $\beta = \vec{p_0}^2$, the Taylor expansion of the inverse of previous expression gives :

$$\frac{1}{r'^4} = \frac{1}{r^4} \sum_{n\geq 0} (-1)^n (n+1) \left(\frac{2\alpha}{r^2}p + \frac{\beta}{r^2}p^2\right)^n \qquad (13)$$

and is defined if and only if :

$$r > (1+\sqrt{2})p \qquad (14)$$

Substituting equation (13) in equation (12), and under the previous condition, we can obtain (cf. appendix 7.1) :

$$K'_{ij} - K_{ij} = \sum_{n\geq 1} p^n \varphi_n^{ij}(\vec{p_0}) \qquad (15)$$

with $\forall n > 0$:

$$\begin{aligned}\varphi_n^{ij} \;=\; & -\frac{1}{\pi r^4} A_j [(\vec{n_i}.\vec{r})(\vec{n_j}.\vec{r})\xi_n \\ & + [(\vec{n_i}.\vec{p_0})(\vec{n_j}.\vec{r}) + (\vec{n_i}.\vec{r})(\vec{n_j}.\vec{p_0})]\,\xi_{n-1} \\ & + (\vec{n_i}.\vec{p_0})(\vec{n_j}.\vec{p_0})\xi_{n-2}]\end{aligned}$$

and

$$\begin{cases} \xi_{-1} = 0,\ \xi_0 = 1, \\ \xi_n = \sum_{q=\lceil \frac{n}{2} \rceil}^{n} (q+1) \binom{q}{n-q} \left(\frac{-4\alpha^2}{\beta r^2}\right)^q \left(\frac{\beta}{2\alpha}\right)^n \end{cases}$$

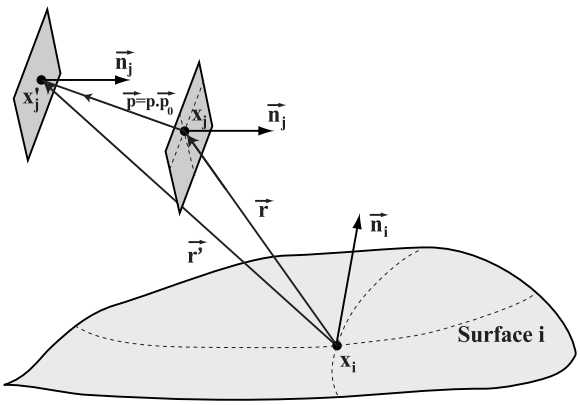

Figure 2. Notations for a translation

We can finally obtain :

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

Equation (16) can compute coefficients ΔK in constant time since n is very small (we choose $n = 6$). Indeed, coefficients φ are recursively defined and depend only on the direction $\overrightarrow{p_0}$, on the degree of approximation n, and on both surfaces i,j and their function k,l. So coefficients φ could be computed once for any elementary directions, as the predefined axes introduced in the next section, and used for any move p in these directions.

3.2.2 Decomposition of complex translations

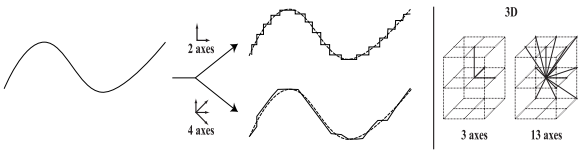

Figure 3. Decomposition of a 2D complex translation and used predefined axes in 3D

The previous approximation is based on the fact that the direction $\vec{p_0}$ is constant. Unfortunately, translations can have very complex shapes, involving frequent changes in the direction $\vec{p_0}$. In order to avoid the calculations of coefficients φ for each small change in direction, we can decompose any complex translation in a succession of small moves, one for each frame, along predefined axes. And since we are dealing with small period of time between each frame, this approximation of the real move is acceptable. Then we will just have to compute the coefficients φ for each used predefined axes. We use a set of thirteen axes in 3D visible in figure 3

3.2.3 Computation of ΔK for a rotation

We will use a far more coarse approximation for the rotation. We start with the equation (2) :

$$K_{ij} = -\frac{(\vec{n_i}.\vec{r})(\vec{n_j}.\vec{r})}{\pi \vec{r}} A_j$$

But this time, we consider that $\vec{r'} = \vec{r}$. This is often true when dealing with small rotations and relatively small surfaces. In this case, we can write :

$$K'_{ij} = \vec{n'_j}.\vec{k_{ij}}$$

with :

$$\vec{k_{ij}} = -\frac{(\vec{n_i}.\vec{r})\vec{r}}{\pi\vec{r}^4}A_j$$

Subtracting it with the previous form factors gives :

$$\Delta K_{ij} = \left(\vec{n_j'} - \vec{n_j}\right).\vec{k}_{ij} = (R - I)\,\vec{n_j}.\vec{k}_{ij} \qquad (17)$$

where R is the rotation matrix. If the surface is planar, we have simply :

$$\Delta K_{ij}^{kl} = \left(\vec{n_j'} - \vec{n_j}\right).\varphi_{ij}^{kl} \qquad (18)$$

with

$$\varphi_{ij}^{kl} = \vec{k}_{ij}^{kl} = \left\langle \iint_{u,v} -\frac{(\vec{n_i}.\vec{r})\vec{r}}{\pi\vec{r}^4}A_j\Gamma^k \mid \Gamma^l \right\rangle$$

For more sophisticated surfaces, we have to decompose the nine coefficients of the matrix $(R - I)$, in a nine coordinates vector M. Then, we can rewrite equation (17) :

$$\Delta K_{ij}^{kl} = \sum_{m=0}^{8} M[m].\varphi_{ij}^{kl}[m] \qquad (19)$$

where $\vec{k}_{ij}^{kl}$ is also a nine coordinate vector defined by :

$$\varphi_{ij}^{kl}[m] = \left\langle \iint_{u,v} \left(\vec{n_j}[m\%3]\vec{k}_{ij}[m/3]\right)\Gamma^k \mid \Gamma^l \right\rangle$$

These approximations are acceptable when (cf. appendix 7.2) :

$$\begin{aligned} \forall r < 1, \quad & sin(\alpha) < \tfrac{rS}{4a} \\ \forall r > 1, \quad & sin(\alpha) < \tfrac{S}{2a} \end{aligned} \qquad (20)$$

where a is the longest distance between the rotation axe and a point of the surface and S a desired quality threshold. If we know the three or nine coefficients φ_{ij}^{kl}, the equation (18) or (19) can compute the variation of form factors in constant time.

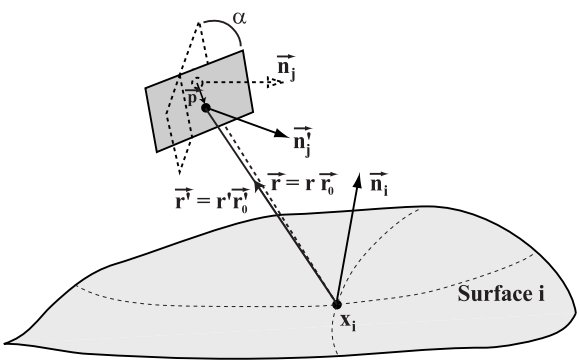

Figure 4. Notations for a rotation

4. IMPLEMENTATION

4.1. General overview

For each new frame of an animation, we start with the previous global illumination solution of the previous frame. We set the unshot radiosity to the two first terms of equation (9) and then use the classical progressive refinement technique. This unshot radiosity is computed, when possible, using the previous approximations, allowing a fast update of the form factors modified. But this is not always possible. Our approximations depend on condition (14) or (20). When they are no more fulfilled, a new form factor has to be computed once again, and its variations deduced. When doing this, all errors done by the previous computations and approximations are erased. We also compute new coefficients φ.

4.2. Dealing with occlusions

The determination of visibility is the central problem of radiosity algorithms. In order to obtain an efficient algorithm, we need some approximations to avoid costly computations of visibility factors. We decide to consider light sources - surfaces with a positive emissivity - apart from the other ones. Indeed, shadows caused by these sources are more important visually than all other occlusion effects. So we decide to use a separate shadow algorithm for all sources, even dynamic ones and to erase visibility from the computation of the form factors and its variations.
Therefore, to handle all the occlusions, we use the following strategy :

- Between two static surfaces, visibility is computed using ray tracing as in [13]
- Between a source and a surface, shadows are computed with a shadow algorithm.
- Between a static surface and a dynamic one, and if none is a source, the visibility factor is computed like two static surfaces as in [13], each time the form factor between them is evaluated.

So, we consider that change in visibility - surface leaving or entering the area of visibility - is taken into account by the form factors and their variations. We also have to point out that the influence of the move of dynamic surface on occlusions of two static surfaces is not considered, except if one of these is a source. This is a coarse approximation and more sophisticated behavior can be achieved to avoid it, for example by using visibility link hierarchy. We choose, for the shadow algorithm of the light source, the one defined by Heckbert and Herf [14]. It involves to store, for each surface, the radiosity function created by each light source.

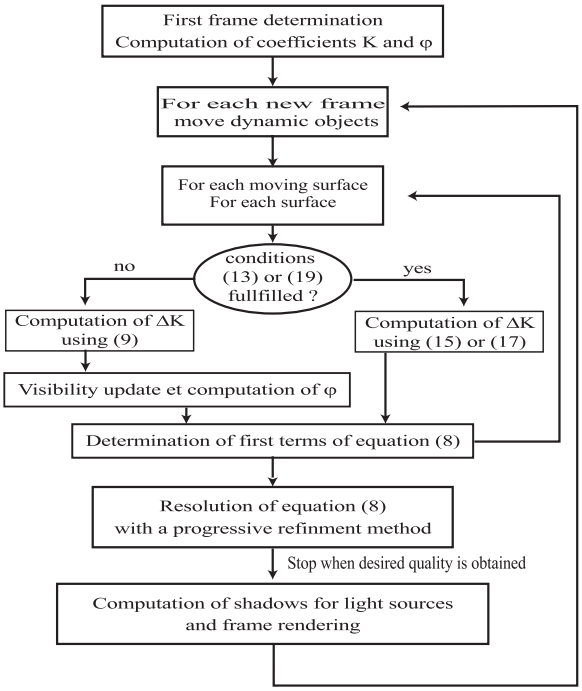

Figure 5. Overview of the algorithm

4.3. Resulting algorithm

The algorithm, depicted by figure 5, starts with the computation of the first frame. Then, for each frame, after having moved the dynamic objects, we determine the two first terms of equation (9). For each pair of surfaces involving a dynamic surface, and depending on the conditions (14) for a translation or (20) for a rotation, variations of form factors could be computed respectively using equations (16) or (18),(19). When these conditions fail, we have to compute the new form factors and the resulting variations using (10). When all pair of such surfaces have been processed, we start the progressive refinement algorithm using, for unshot radiosity, the two first terms of equation (9). When a desired quality is obtained for this frame, it can be rendered using, for all light sources, a shadow algorithm (e.g. the one of Heckbert and Herf [14]). The functions base used is the legendre polynomial functions base.

5. RESULTS

The main computation time in the radiosity algorithms, except for visibility testing, is due to determination of pseudo form factors and it is espeacially true when dealing with higher order functions algorithms. Our algorithm avoids computing them for each frame thanks to the computations of coefficients φ, which take into account the temporal coherence of the moves. Moreover, only necessary energy differences needed to obtain the new global illumination are computed since we use a progressive method for each position (normal progressive method) and also between each position (our temporal progressive method). So we save time in both improving convergence and avoiding form factors computations. Notice also that moves do not have to be known in advance.

We present in this article some results and images from animation sequences[1] computed with a 500 MHz processor and a common PC graphic card. Table 1 shows computation times for global illumination determination and the benefits of our method compared to the classical one (a progressive refinement algorithm for higher order functions). This is done for three animations of three different scenes illustrated in figure 6. The first animation (figure 6.a on the left) presents two moving surfaces, with one light source. The second animation (figure 6.b in the center) handles the case of a light in rotation when the third and the forth one (figure 6.c and d on the right) present moving light in more complex scenes. Notice also that sharp shadow lines come from the fact that light sources are planar.

Scenes	6.a	6.b	6.c	6.d
Classical time/frame	2.12 s	342 ms	639 ms	2 s
Our time/frame	1.1 s	45 ms	181 ms	870 ms
Benefits	48 %	87 %	72 %	56 %

Table 1. Results for scenes of figure 6

We also show in the figure 7 the relative errors between our animation frames and the correct frame for the scene of figure 6.b. It can be seen that our method does not affect the quality of the animation. In fact, thanks to the erasing of errors done when approximation conditions are no more fulfilled, we can control the overall errors. Our method takes effectively into account the global illumination. For instance, in scene of figure 6.a, when the blue panel pass in front of the light, the wall is correctly rendered in blue. We point outs also that this scene is a worst case because the blue panel is very close of the light and it crosses its area of visibility.

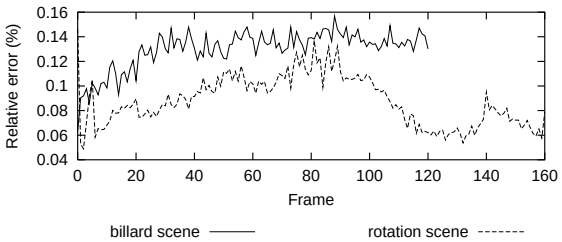

Figure 7. Errors of two animations

6. CONCLUSION

In this paper, we have presented a new algorithm able to compute long radiosity animations with any mov-

[1] Available in http://www-igm.univ-mlv.fr/~biri/indexCA.html

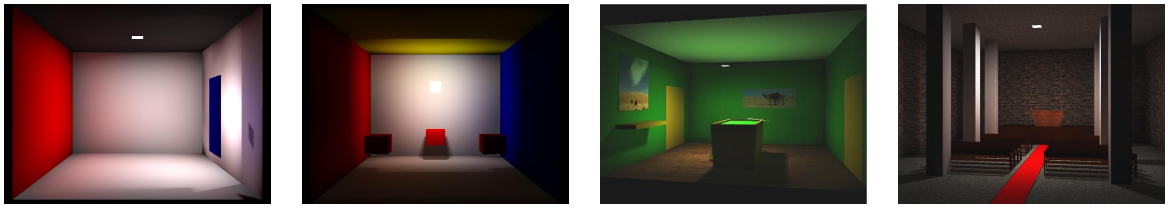

Figure 6. (from left to right) a) The simple scene b) Rotation scene c) Billard scene d) Cathedrale scene

ing surfaces - and even light sources. Timesaving is obtained, in each frame of the sequence, by avoiding the computations, for each moving object, of their new form factors. Instead, this algorithm focuses on the variations in illumination rather than computing the new global illumination solution. The form factors variations, depending on the moves, allow to handle intensive temporal coherence.

Efforts should be paid now on efficient shadow algorithms allowing fast and accurate soft shadows for area sources in a dynamic environment. In order to speed up computation time, we can also use clusterisation. For example, 3DS model, consisting of many small triangles and that can not be considered like one surface, could be embedded in a cluster which can exchange illumination with other surfaces. This will also speed the shadow determination. We hope finally to combine efficiently this progressive approach with hierarchical algorithms that minimise the number of radiosity exchanges between surfaces.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] C. Damez, N. Holzschuch, and F. Sillion. Space-time hierarchical radiosity with clustering and higher-order wavelets. In *Eurographics 2001 Short Presentations*, pages 35–42, september 2001.

[7] J. Dorsay, F. Sillion, and D. Greenberg. Design and simulation of opera lighting and projection effects. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25(4), pages 247–257, 1991.

[8] Y. Dupuy, F. Lavignotte, and M. Paulin. Visibilité et Radiosité Interactive. In *12eme journée de l'AFIG. AFIG'99 Conference Proceeding*, pages 41–50, Nov. 1999.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

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

[17] 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.

[18] 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.

[19] K. Nielsen and N. Christensen. Real-Time Dynamic Relighting of Virtual Environments. In *Journal of WSCG*, volume 10(2), pages 325–331, Feb. 2002.

[20] P. Schoder. Numerical Integration for Radiosity in the Presence of Singularities. In *4th Eurographics Workshop on Rendering*, pages 177–184, 1993.

[21] 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.

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

[23] 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.

7. APPENDIX

7.1. Computation of ΔK

Using the following notations :

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

we have

$$K'_{ij} = -\frac{A_j}{\pi r^4}\left(a+bp+cp^2\right) \sum_{n\geq 0}(-1)^n(n+1)\left(2\frac{\alpha}{r^2}p+\frac{\beta}{r^2}p^2\right)^n$$

or

$$K'_{ij} = K_{ij} - \frac{A_j}{\pi r^4}(bp+cp^2) - \frac{A_j}{\pi r^4}(a+bp+cp^2) \sum_{n\geq 1}(-1)^n(n+1)\left(2\frac{\alpha}{r^2}p+\frac{\beta}{r^2}p^2\right)^n$$

We want to extract p of the last term :

$$\begin{aligned} &\textstyle\sum_{n\geq 1}\frac{(-1)^n(n+1)}{r^{2n}}\sum_{k=0}^{n}\binom{n}{k}(2\alpha p)^{n-k}\beta^k p^{2k} \\ &= \textstyle\sum_{n\geq 1}\frac{(-1)^n(n+1)}{r^{2n}}\sum_{k=0}^{n}\binom{n}{k}(2\alpha)^{n-k}\beta^k p^{k+n} \\ &= \textstyle\sum_{n\geq 1}\sum_{l=n}^{2n}\frac{(-1)^n(n+1)}{r^{2n}}\binom{n}{l-n}(2\alpha)^{2n-l}\beta^{l-n}p^l \\ &with\ \ l=n+k \\ &= \textstyle\sum_{l\geq 1}p^l\sum_{n=\lceil\frac{l}{2}\rceil}^{l}\frac{(-1)^n(n+1)}{r^{2n}}\binom{n}{l-n}(2\alpha)^{2n-l}\beta^{l-n} \\ &= \textstyle\sum_{l\geq 1}p^l\ \xi_l \end{aligned}$$

Exchanging l and n, ξ have the expression :

$$\forall n>0,\ \xi_n = \sum_{l=\lceil\frac{n}{2}\rceil}^{n}(l+1)\binom{l}{n-l}\left(\frac{-4\alpha^2}{\beta r^2}\right)^l\left(\frac{\beta}{2\alpha}\right)^n$$

and

$$(bp+cp^2)+(a+bp+cp^2)\sum_{n\geq 1}p^n\xi_n = \sum_{n\geq 3}p^n(a\xi_n+b\xi_{n-1}+c\xi_{n-2})$$

We have then

$$K'_{ij} - K_{ij} = \sum_{n\geq 1}p^n\varphi_n^{ij}(\vec{p_0})$$

with $\forall n>0$:

$$\begin{cases} \varphi_n^{ij} &= -\frac{A_j}{\pi r^4}(a\xi_n+b\xi_{n-1}+c\xi_{n-2}) \\ \xi_{-1} &= 0 \\ \xi_0 &= 1 \end{cases}$$

7.2. Limit of the approximation for rotation

If we denote Δ the expression of (13), we have :

$$K'_{ij} - K_{ij} = \frac{(\vec{n_j}.\vec{p})(\vec{n_i}.\vec{r})}{\pi r^4} + \frac{(\vec{n_j}.\vec{r})(\vec{n_i}.\vec{p})}{\pi r^4} + \frac{(\vec{n_j}.\vec{p})(\vec{n_i}.\vec{p})}{\pi r^4} + \frac{(\vec{n_j}.\vec{r'})(\vec{n_i}.\vec{r'})}{\pi}\Delta$$

and

$$\left\|K'_{ij}-K_{ij}\right\| \leq \frac{2p}{\pi r^3}+\frac{p^2}{\pi r^4}+\frac{(r+p)(r+p)}{\pi}\Delta$$

We consider now that $\xi=\frac{p}{r}<1$ and is very small so we can neglect terms in ξ^2. Then :

$$\left\|K'_{ij}-K_{ij}\right\| \leq \frac{1}{\pi r^2}\left|2\xi+\left(1+\frac{2\xi}{r}\right)\left[-\frac{4\xi}{r}\right]\right|+o(\xi^2)$$

and finally :

$$\left\|K'_{ij}-K_{ij}\right\| \leq \frac{2\xi}{\pi r^2}\left|\frac{r-2}{r}\right|$$

when $r>1$, we have $\frac{r-2}{r}<1$, the relative error χ is :

$$\chi \leq 2\xi = \frac{2p}{r}$$

and when $r>1$, we have $\frac{2-r}{r}<\frac{2}{r}$, the relative error is :

$$\chi \leq \frac{4p}{r^2}$$

With $p < a\ sin(\alpha)$ where a is the largest dimension from the rotation axe and a surface point, both previous equations lead to the conditions (20).