THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION
*J. Visual. Comput. Animat.* 2001; **12:** 93–106 (**DOI:** 10.1002 / vis.248)

# *Animating radiosity environments through the Multi-Frame Lighting Method*

*By Gonzalo Besuievsky* and Xavier Pueyo*

***Realistic rendering animation is known to be an expensive processing task when physically based global illumination methods are used in order to improve illumination details. This paper presents the Multi-Frame Lighting Method, an efficient algorithm to compute animations in radiosity environments. The method, based on global Monte Carlo techniques, performs the lighting simulation of groups of consecutive frames in a single process. All frames computed have the same accuracy as if they were computed independently while a significant high speed-up is achieved. Results show that the method it is an interesting alternative for computing non-interactive radiosity animations for moderately complex scenarios. Copyright © 2001 John Wiley & Sons, Ltd.***

*Received: January 2000; Revised: December 2000*

KEY WORDS: realistic animation; Monte Carlo; radiosity

## Introduction

High-quality realistic animation is often a desired objective in image synthesis. The accurate computation of the illumination plays a very important role in the improvement of realism on simulated sequences. For this reason, global illumination models are required when computing realistic animations. These models have in general a high computational cost when dealing with complex scenarios where we want to simulate illumination details like soft shadows, diffuse interreflections or more complex lighting phenomena involving caustics or participating media. For animations, usually composed of sequences of lots of images, this problem grows, leading to large amounts of computation if each frame is treated independently. Thus, efficient solutions are searched in order to reduce the computation cost. Considering that in most cases changes in illumination between consecutive frames are usually smooth as a consequence that animation changes are also smooth, a natural possibility for an efficient approach is to improve global illumination models in order to make use of this source of coherence. For example, a significant computation time can be saved by reusing illumination already computed that is invariant in an interval of time within the animation.

The radiosity method, introduced in Goral *et al.*,[1] allows interactive walkthroughs since it is a view-independent global illumination solution. For this reason, it is an attractive method to be used both in animations and in interactive applications. Many works with a focus on this last domain have been proposed. Strategies like progressive refinements methods[2–4] or hierarchical radiosity[5–7] have been developed with the goal of quickly updating changes in illumination for a given demand. In the case of non-interactive radiosity animations,[8–12] where all time-dependent parameters can be known a priori, requirements are not the same as in the interactive case: there is not a high compromise on speeding updates of single frames. However, a large number of frames must be computed, maintaining a high-quality level of accuracy. Although in most cases interactive techniques could be adapted to animation purposes, a specific approach for animations can extract more benefits.

In this paper we present the Multi-Frame Lighting Method, a Monte Carlo-based method that proposes an efficient solution for animated radiosity environments by processing many frames of a given animation at once, within a single simulation of light transport. A first approach of the method was presented with early results in Beseuievsky and Sbert:[13] we present here the

*Correspondence to: G. Besuievsky, Institut d'Informàtica i Aplicacions, Dept IMA, Campus Montilivi, E-17071 Girona, Spain.

fully developed algorithm and its analysis. Results show that it is a good alternative for computing radiosity animations for moderately complex scenarios. The speed-up of the method, that is, the efficiency comparing the method to an independent per-frame processing, could ranges from about 7 to 25 without loss of quality in the simulation, depending on the parameters of the animation. We analyze the efficiency for different kinds of scenarios and animating parameters.

In the next section an overview of the work done in related topics is presented. The Multi-Frame Lighting Method and its analysis are presented in the third section. The fourth section describes the implementation and usability of the method. Results are shown in the fifth section. A final discussion relating our approach to previous techniques is given in the sixth section and future work is described in the final section.

# Background

## Radiosity in Animated Environments

Animated radiosity environments can be defined as environments where the number of objects, their shape, location or surface attributes, as reflectance or emittance, may change in time.[5] Techniques involving this kind of environment can be classified according to its usability into two main groups:

1. Interactive (or near real-time) algorithms:[2–7] For a required change in the environment the illumination must be updated interactively without loss of important illumination information. A high compromise in time response is needed. Most of previous work focuses on this kind of algorithm because of its wide range of uses. Some examples of application are: architecture CAD systems for indoor design, virtual studios, virtual environments and driving simulators.
2. High-quality rendering animation algorithms:[8–12] A large number of frames must be computed with a high level of accuracy in order to simulate a realistic animated sequence. All animated parameters may be considered to be known in advance. In this case, the level of accuracy of the lighting simulation is very important: a small discontinuity in quality could be enough to lose the realism desired in the illusion. Application examples are high-quality realistic animations for movies, and animated sequences to be blended with live action video and films.

Although the problem is similar in both cases, proposed solutions are often driven by the kind of requirements of the applications. In the interactive case there is a high compromise in the time response. Also, the trade-off between quality and speed is frequently desired to be tunable. Progressive refinement strategies are widely used to attain these requirements. However, in high-quality rendering animation, where all dynamic parameters are known in advance there is a highest compromise in the quality of the simulation.

In some cases, interactive techniques could be adapted to animation, but a specific approach for this kind of application can give more benefits. Considering that all changes in the environment are previously known, invariants in illumination may be detected more easily than in the interactive case and, as a consequence, coherence may be exploited better. Another question that could discourage the use of interactive techniques for animation is that in general they provide a fast solution to update single changes but not for a set of sequential images (an exception is the method presented in Drettakis and Sillion[6]). We review the most important contributions of animated radiosity techniques.

The first approach for animation in radiosity environments imposes some restrictions to the problem in order to simplify the solution. Baum *et al*.[8] based their algorithm on an extension of the traditional radiosity algorithm where a fixed point of view is assumed. With the aim of exploiting temporal coherence the invariance of the geometrical relationship between static surfaces and a fixed observer is detected, reducing in this way the number of form factors to be recomputed. Such detection is managed through an auxiliary swept volume, which encloses the moving path of the dynamic object. In Forsyth *et al*.,[9] a hierarchical radiosity solution is sought by updating links between surfaces depending on the movement of objects. Nimeroff *et al*.[10] presented a range image-based framework for animated global illuminated environments. The system strategy consists of computing a set of base view images from which the whole animation is obtained by interpolation. For the lighting computation, direct and indirect illumination are split into different independent processes in order to improve spatial and temporal coherence. Although this image-based approach can provide a partially

independent view solution it is still restricted to a base view-space pre-computed and a complete change in the animated camera would require the computation of a new base view. An improvement of this method is that the global illumination solution it provides is not restricted, as it usually is, to diffuse reflectance surfaces but can also account for glossy surfaces.

Recently, two novel methods for animation were introduced: a space-time solution for a hierarchical radiosity refinement that refines links in both space and time,[11] and a frame-to-frame coherence approach based on a two-pass step radiosity solution.[12] Both methods based the global illumination solution on an extension of hierarchical radiosity over time. This is an interesting avenue because it is the best way to deal with error control due to surface discretization in the light transport, but on the other hand they suffer from huge memory requirements, thus restricting the methods to use in scenarios of limited size.

Given that the main goal of animated techniques is the acceleration of the computation of a whole sequence, we examined the speed-up obtained in previous work. Surprisingly, although in most cases highly coherent animations are tested, not very high speed-ups are reported: 2 to 2.5 in Reference 11, about 4 in Reference 9, 2.3–7.3 in Reference 12 or about 8 in Reference 10. In References 5 and 11 experiments to demonstrate the high degree of illumination coherence in scenes when performing smooth changes in an object's position were performed. According to Damez and Sillion,[11] for a simple short animation a percentage ranging between 50% and 70% of the links in a hierarchical radiosity solution does not change during the whole sequence, meaning that only a small percentage of illumination changes. Considering this assertion we believe that speed-up can still be improved.

## Managing Visibility Changes

The visibility computation is a very important task in illumination algorithms. For animated environments, visibility interaction between different kinds of surfaces (static and dynamic) must be efficiently detected and computed. The visibility between any pair of surfaces could potentially be affected by modification in geometry in the scene. The identification of visibility changes is in general a hard task and has been tackled through different auxiliary structures and algorithms in previous work. Drettakis and Sillion[6] used an auxiliary shaft culling structure[14] to quickly identify and update important link changes in their line-space approach. Fast change identifications are performed by hierarchically descending in the line-space hierarchy. In Müller and Schoeffel[4] a dynamic shadow-form-factor-list structure is used to easily detect important occlusions in direct illumination. In Shaw[5] a motion bounding box volume of dynamic objects is built and used in conjunction with a clipping algorithm to quickly detect changes in visibility. Other structures that can be used to update visibility changes are the *Visibility Complex*[15,16] and the *Visibility Skeleton*,[17] developed with the aim of encoding all visibility information in a three-dimensional space.

Our approach is based on global Monte Carlo methods.[18–21] These methods allow encoding of all visibility information implicitly in a relatively simple way by traversing the scene's space through sets of lines.

## Global Monte Carlo

Monte Carlo methods are frequently used in light transport calculation when a physically based lighting simulation is required. By treating the simulation in a stochastic fashion we can better deal with complex scenes and complex light transport phenomena. First uses of Monte Carlo methods in radiosity were proposed in References 22 and 23. Later improvements[24–27] show the usability of these kinds of approach as an alternative to deterministic methods. They provide a reliable form factor computation while no explicit storage of the form factor is needed. Also, a quickly 'correct' solution showing high-order interreflections can be obtained. On the other hand, they suffer from noticeable noise artifacts until the necessary amount of samples is traced, which could lead to time-consuming processes.

Two branches of Monte Carlo radiosity algorithms can be distinguish according to the way lines are cast: local Monte Carlo[22–25] or Global Monte Carlo.[19–21,28] While local Monte Carlo techniques cast rays from surfaces, global Monte Carlo cast lines that are independent from surface positions; these lines are called global lines.

Based on integral geometry concepts, global approaches simulate light transport computation by generating a uniform global density of lines within the scene. The number of lines crossing a polygon area submerged in this kind of density is constant no matter the position and orientation of the polygon. Since the distribution of lines intersecting a surface is the cosine distribution,[19] these lines can be used to compute

diffuse interreflection light transport. The most used ways to build global densities are, by taking two random points in a bounding sphere of the whole scene (Figure 1), or by getting parallel bundles of lines from random directions. Each global line traverses the whole scene and its intersection with the scene's surfaces is computed. A visibility list, which faces all pairs of mutually visible patches, is created by sorting the list of intersections. Light energy transfer is computed using these visibility lists.[26]

## The Multi-Frame Lighting Method

Consider a dynamic object moving in an environment. As described in the previous section, visibility between any kind of objects may change for each new position of the moving object. In order to compute efficiently all these changes we would like to access visibility information for different frames at the same moment. Global methods give us this possibility. An interesting feature of these methods, in static scenes, is that all intersections of a single global line with the surfaces of the scene can be used to compute light energy transport. If we have dynamic surfaces moving in the scene, then global lines would also be valid for the different positions in time of such surfaces. This is valid from integral geometry results[19,29] if the range position of the motion lies inside the bounding sphere of the scene.

Figure 2 illustrates an example of the usability of a global line when dealing with dynamic objects: a change in visibility between two static objects caused by an occlusion of a dynamic object can be managed through a visibility list that contains visibility information with and without occlusion. Consider the inner walls of the room in the figure; at a given time $t_1$ the cube is hiding patch P from Q, whereas for instant $t_2$ it is visible. All this information can be encoded in a visibility list obtained from the intersection of the global line with the scene. The usability of global lines to compute light energy when surfaces are at different positions is used to develop the Multi-Frame Lighting Method (MFLM).

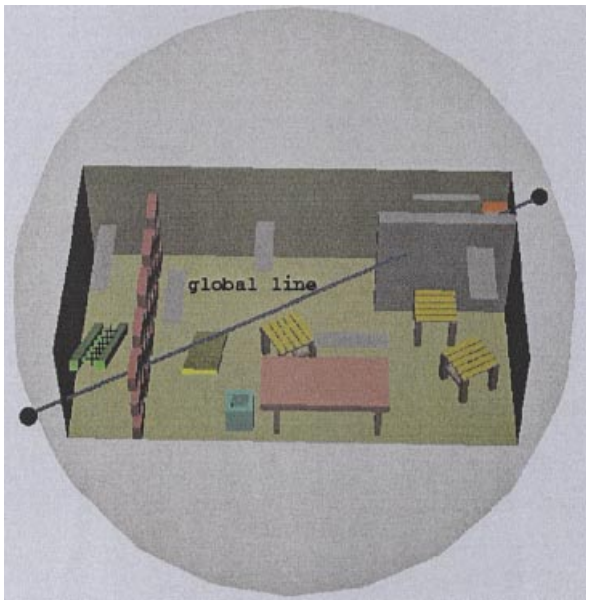


*Figure 1. A global line can be built from two random points laying in a bounding sphere of the scene.*

### The Algorithm

The method works in a multi-processing task strategy: given an animated sequence description of the environment it computes the radiosity solution of all frames in a single process. All surfaces of the environment are considered as having an implicit discretization into patches, but it may be generalized for an adaptive discretization approach. The method consists of the three following steps: pre-processing, lighting computation and visualization.

**Pre-Processing Step.** In order to prepare the animated model for the lighting computation we process the animation storing the dynamic object's information for all frames. In the further lighting

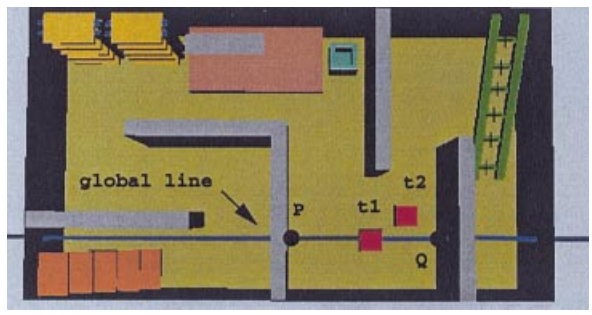


*Figure 2. The global line traversing the object at position A at time $t_1$ is also valid for the object at B at time $t_2$. A change in visibility between patches P and Q can be detected by processing the visibility list.*

computation step the model is going to be intersected for all frames of the sequence. The aim of this pre-processing step is to obtain an animation model where we can have quick access to any dynamic information, like the position of an object, at any time.

Two different approaches can be used for this step: store the dynamic object's position for all frames or store only the final matrix transformation for all frames. In the first case a super-scene containing all objects at all time positions is generated and the intersection procedure will follow a standard process with a larger scene. In the case of storing the matrix transformations, the ray must be transformed by the inverse matrix transformation of the motion at each frame and tested for intersection with the object at the original position. We prefer this last one because it optimizes the amount of memory required. Thus, our pre-processing step consists of the computation and storing of the forward and inverse transformation for each group of objects moving with the same motion specification.

**Lighting Computation Step.** We compute the illumination for all frames independently in a single simulation process. This process includes the following tasks:

- *Intersection*: Cast global lines and intersect the scene at all frame times. All objects, static and dynamic at all frame positions of the animation are intersected (Figure 3). Intersections obtained for a given global line are stored in two temporary lists: one for the static objects and another for the dynamic ones (Figure 4 (top)). Each element of the dynamic list is labeled with the number of the frame it belongs to. These lists are used to build the visibility lists.
- *Visibility lists*: From each global line different visibility lists, one for each frame, are built. This is done by sorting and merging the static and the dynamic lists of intersections obtained for the corresponding frame. Identification of the frames is done using the label number attached to the dynamic items in the previous intersection operation. Figure 4 illustrates the list operations following the example of Figure 3: from the static and dynamic lists obtained, four visibility lists are built, one for each frame.
- *Light energy transfer*: The visibility lists are used to compute light energy transport between visible pairs of patches. The global multi-path[26] or the global transfer[30] algorithms can be used for this purpose. We maintain an independent radiosity solution for each frame of the sequence.

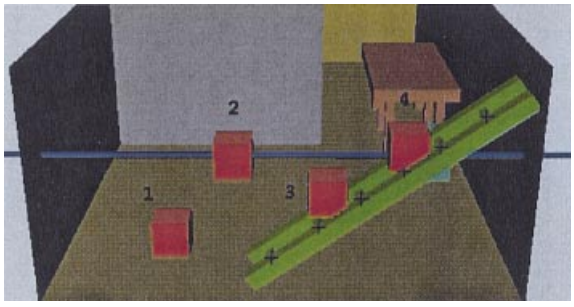


*Figure 3. Global lines are cast to intersect static objects and dynamic objects (the cube in this example) at all their instances.*

**Visualization Step.** Once the radiosity animation is computed the sequence can be played for a given camera description by projecting the radiosity solution for each frame. Hardware z-buffer is used for this purpose.

## Analysis

**Storage.** The most significant memory requirements of the method are the power energy counters needed to perform the simulation. A complete radiosity solution for each frame is stored in the method. Using the global multi-path algorithm[26] for the lighting computation we need three spectral counters for each patch of the scene (the exit light power, the unshot power and the accumulated power); then, for the multi-frame we will need all these counters times the number of frames of the sequence.

**Cost.** We study the cost of the MFLM with the aim of analyzing its efficiency. Suppose we want to compute an animation sequence along $k$ frames within a scene composed by $n$ polygons where $n_s$ of them are static and $n_d$ are dynamic. We consider that the geometry of the scene is organized in a hierarchical structure for its intersection acceleration. For a scene composed of $n$ surfaces, the complexity of finding intersections for a given line is $O(\log n)$. In the development of our analysis we name as $C_i$ ($i=1$, 2, …) all constants of proportionality. Each time a global line is cast the MFLM intersects all static surfaces and for each frame all dynamic surfaces. We also organize the bounding boxes of all frames by

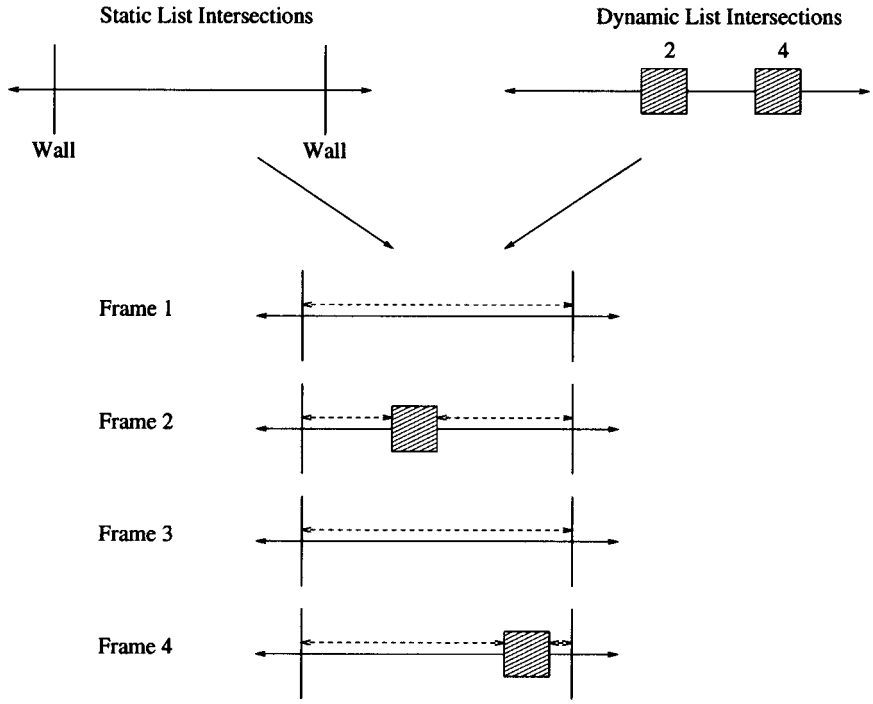


*Figure 4. Following the global line example of Figure 3, a visibility list for each frame is built by merging and sorting all static and dynamic intersections. Light energy transfers are computed using these lists.*

clustering neighbor frames in a hierarchical bounding box structure. Under these conditions, the cost of intersection of a single line with the dynamic environment can be expressed as

$$C_{int} = C_1 \log n_s + C_2 \log k \log n_d \tag{1}$$

Let us analyze the cost of processing the lists of intersections we obtain. Suppose that for each line we intersect an average number of static polygons $n_{is}$ and an average number of dynamic polygons per frame $n_{id}$. The cost of sorting intersections and merging the lists to obtain the visibility lists is

$$\begin{aligned} C_{lists} = {} & C_3 n_{is} \log n_{is} + C_4 k n_{id} \log n_{id} \\ & + C_5 k (n_{is} + n_{id}) \end{aligned} \tag{2}$$

The first term of this equation represents the cost of sorting the static intersections, the second, the cost of sorting the dynamic intersections for each frame and the third term the cost of merging the sorted static list and the sorted dynamic list for each frame. Suppose we cast $N$ lines, only a fraction of these lines will intersect the dynamic part of the scene. Thus, terms in equation (2) are not proportional to $N$ in the same way depending on whether they represent a static or a dynamic part. The number of lines intersecting the dynamic part of the scene can be predicted using results of integral geometry: if the area covered by the bounding box of the motion volume is $A_d$ and we cast lines from a surrounding body of area $A$, the number of lines crossing the motion volume is $\frac{2A_d}{A}N$. Therefore, the cost of processing the lists when $N$ lines are cast is

$$\begin{aligned} C_{Nlists} = {} & N C_3 n_{is} \log n_{is} \\ & + \frac{2A_d N}{A} (C_4 k n_{id} \log n_{id} \\ & + C_5 k (n_{is} + n_{id})) \end{aligned} \tag{3}$$

Then, the total cost of the method when we cast $N$ lines is

$$\begin{aligned} C_{tot} = {} & N(C_1 \log n_s + C_2 \log k \log n_d \\ & + C_3 n_{is} \log n_{is}) + \frac{2A_d N}{A} (C_4 k n_{id} \log n_{id} \\ & + C_5 k (n_{is} + n_{id})) \end{aligned} \tag{4}$$

In a single simulation we obtain $k$ frames solutions; thus, the total cost per frame of the animated sequence is

$$C_{totFr} = N\left(\frac{C_1 \log n_s + C_2 \log k \log n_d + C_3 n_{is} \log n_{is}}{k} + \frac{2A_d}{A}(C_4 n_{id} \log n_{id} + C_5(n_{is} + n_{id}))\right) \quad (5)$$

The most important aspect of this equation is that the cost per frame decreases as the number of frames of the animation grows. Figure 5 shows the theoretical curve of the cost per frame as a function of the number of frames. Although it is clear from this curve that the method is more efficient if more frames are processed together, we can also note that the cost is low bounded. That is, for a high number of frames the time per frame will remain almost constant, meaning that no increase in efficiency is observed. In order to evaluate the efficiency, we define the speed-up of the method for a given animation as the ratio between the cost of processing a single frame and the bound of cost per frame obtained with the method.

### Direct Lighting Optimization

Direct lighting has a very important incidence in the result of the lighting simulation. Important illumination issues such as soft shadows must be accurately computed to improve realism. Global Monte Carlo methods have a significant increase in efficiency if the initial power is well distributed along surfaces on the scene.[26] For the MFLM, as it is based on global methods the same consideration is valid. We propose two different approaches for direct lighting optimization in the MFLM, distinguishing whether light sources are static or dynamic.

If lights are static the first-shot-from-lights step proposed by Sbert *et al.*[26] for static scenes can be extended for animated scenes.[13] Direct illumination is distributed by casting lines from sources and intersecting, as in the main algorithm of the MFLM, the dynamic objects at all instants of time.

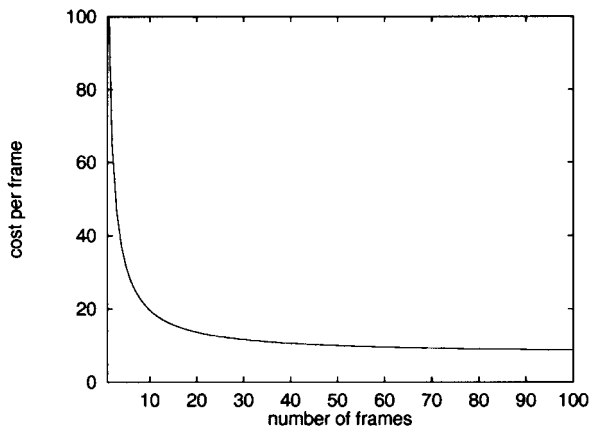


*Figure 5. Theoretical cost per frame vs. number of frames.*

Animations where light sources are in motion are more time demanding because less illumination coherence can be exploited as a consequence that different distributions of primary energy are required for each frame. The MFLM as presented in 'The Algorithm' section above is not restricted to dynamic light sources but a specific algorithm could optimize the method for this kind of environment. If light sources are in motion direct lighting is efficiently distributed for all frames by casting global lines within the motion volume of the sources. The complete algorithm and its analysis is presented in References 31 and 32.

We must mention that in the analysis of the cost ('Cost' section above) we did not include the direct lighting optimization described here. However, the behavior of the cost of this step is similar to the general MFLM;[32] therefore, the same conclusions about the efficiency can be extracted.

## Implementation and Usability

The MFLM is implemented over SIR,[33] a framework that provides programmers with a basic set of object-oriented components to be extended for specific global illumination algorithms.

Once the sequence is completely designed, classic animation software acts by transforming all dynamic objects and camera parameters to compute each frame and stores the resulting rendered images. A different pipeline is used within the MFLM (Figure 6). The scene environment and the animation description are taken as input for the method. Any animation tool that provides a way to record an animation description could be used. We use Maya (A/W) to design the animation. After the lighting simulation is finished, a radiosity representation of the whole animation is recorded into a file. This representation is completely view independent, conserving in this way a basic property of static radiosity solution, for animations. This means that any animated camera description can be used to visualize and obtain the final images of the animation. For this last step we use hardware z-buffer, which has a relatively low cost. Anti-aliasing processing and motion blur effects (see 'Other Issues and

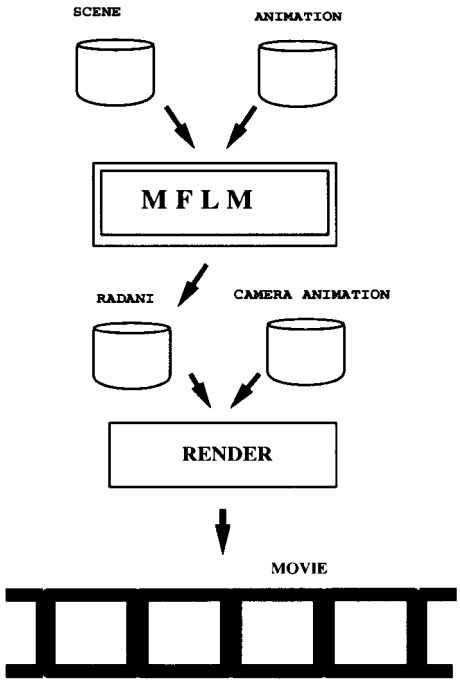


*Figure 6. Animation processing in the MFLM. Once the lighting simulation ends, a view-independent radiosity representation of the animation (RADANI) is recorded and rendered from an animated camera.*

Future Work' below) may also be performed at this final rendering step.

In the 'Analysis' section above it was shown that the MFLM has a bound of efficiency. That is, after a number of frames $k_c$ (see Figure 8), no increase in efficiency will be observed by computing more frames together. This number depends on the animation parameters and can be estimated in a fast way by performing the method with a low number of lines. Our approach for the best usability of the MFLM is to split the animation in sub-animations of $k_i$ frames, where $k_i$ is greater than $k_c$, and to process independently each sub-animation. Following this strategy the computation of the whole animation can be distributed among many processors.[32]

# Results

## Animation Tests

We can see from equation (5) that the efficiency of an animation depends on several parameters: the proportion of static and dynamic surfaces ($n_s$ and $n_d$), the average number of intersections in the scene ($n_{is}$ and $n_{id}$) and the bounding area covering the motion volume of the dynamic objects. We test the MFLM for

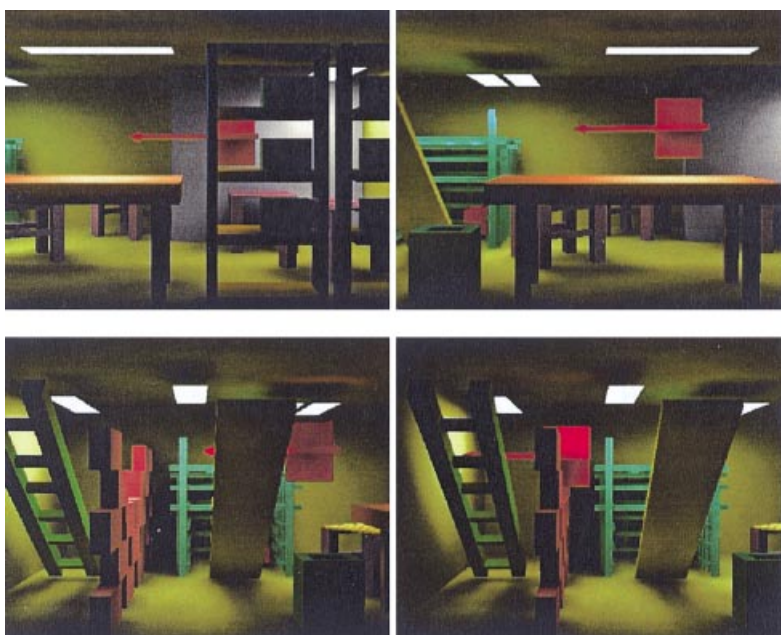

*Figure 7. Four frames of the animated sequence of Test 1. The environment is composed of 1372 polygons. Only the arrow, composed of 44 polygons, is animated.*

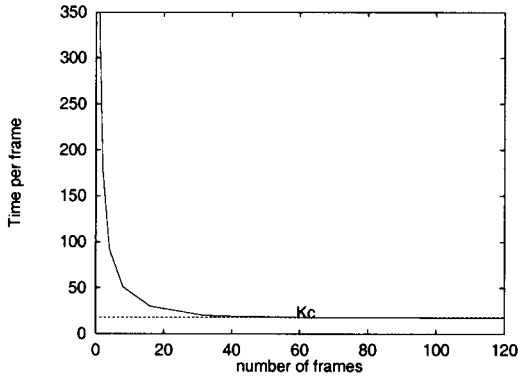


*Figure 8. Time per frame vs. number of frames for the animation Test 1. The time per frame decreases as more frames are processed together until a bound of about 60 frames (Kc).*

different kinds of animated scenarios, varying these parameters in order to confirm the theoretical behavior of the method. We use the speed-up as a measure of the performance of the method: the time processing of a single frame divided by the time per frame obtained with our method. Figures 7, 9, 10 and 11 show frames of the sequences used for the tests. Animations shown in this paper can be seen in MPEG-1 format at the following site: http://ima.udg.es/~gonzalo/animations/mflm.html. All executions were measured on on a PC-Linux Pentium III at 450 MHz.

**Test 1: Cost per Frame.** An arrow is animated in a scene containing 1372 polygons (Figure 7). We execute the sequence for different numbers of frames. Table 1 shows the times obtained; the curve time per frame vs. number of frames is plotted in Figure 8. The expected behavior of the theoretical curve of Figure 5 can be appreciated: the cost (measured as the time per frame) decreases as the number of frames processed together grows. For this animated scene the low bound of the time per frame happens for near 60 frames (label as Kc in Figure 8), giving a speed-up of 18.

We tested a longer sequence in the same environment with more dynamic objects. The sequence, composed of 180 frames, has 1185 surfaces where 131 are dynamic (Figure 9). Following the strategy

*Figure 9. Four frames of the second animated sequence of Test 1. The scene is composed of 1316 polygons: 1185 static and 131 dynamic.*

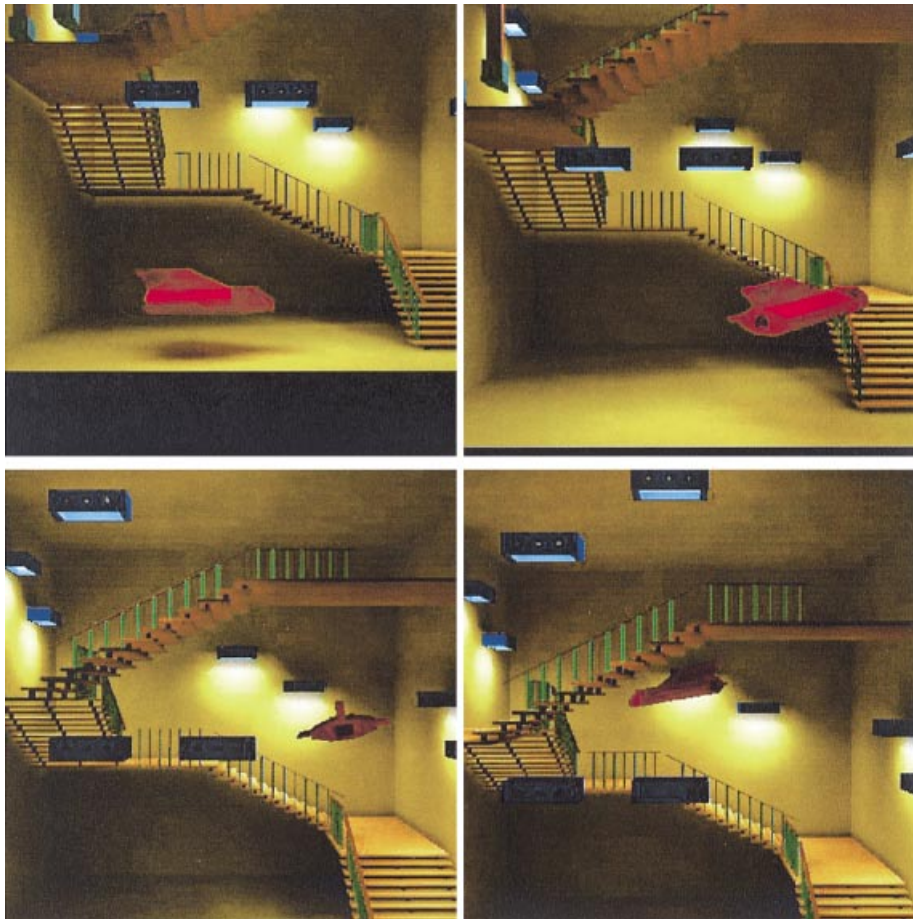

*Figure 10. Four frames of the animation Test 2. The scene contain 3806 polygons, where 126 are dynamic.*

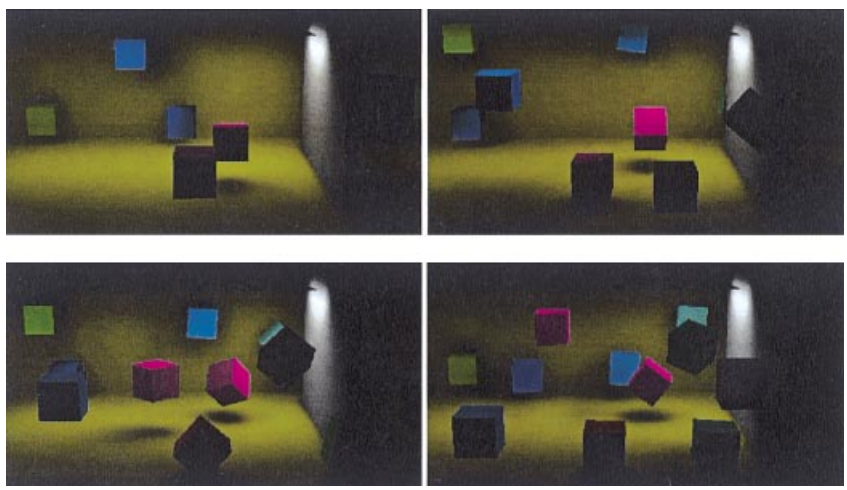

*Figure 11. Four frames of the animated sequence of Test 3. The environment is composed of 10 cubes, each one with different motion path.*

| Number of frames | Time per frame (s) |
|---|---|
| 1 | 350 |
| 2 | 179 |
| 4 | 92 |
| 8 | 51 |
| 16 | 30 |
| 32 | 20 |
| 60 | 18 |
| 120 | 17 |

**Table 1. Time per frame results for the animation Test 1. The same animation is computed for different numbers of frames**

described in 'Implementation and Usability' above we split the sequence in six groups of 60 frames, each groups entailing a completely independent simulation. The time per frame and the speed-up obtained for each group are shown in Table 2. The total execution time for the sequence is 77 minutes and the average speed-up 16.8. The amount of memory required in the execution for each group of frames was about 40 Mb. Close to $10^7$ lines where cast for the simulation for a discretization of the scene into 23,000 patches.

**Test 2: Length of Motion Path.** In this scenario (Figure 10) only a flying plane is dynamic. Consider the volume sweep by this object along its movement. For this test, we vary the length of the motion path, getting always the same number of frames (120). Consider the motion volume described by this object along its movement. The factor $\frac{A_d}{A}$, the ratio between the area of the motion volume and the area of the bounding box of the whole scene (see equation 5), will be affected for such motion path variation. The area of the motion volume ($A_d$) increases as the length of the motion path is longer. Table 3 shows time per frame and speed-up resulting for each ratio $\frac{A_d}{A}$. The speed-up increases as factor $\frac{A_d}{A}$ is reduced, corresponding to animations with shorter path.

| Range of frames | Time per frame (s) | Speed-up |
|---|---|---|
| 1–13 | 26 | 17.2 |
| 31–60 | 24 | 17.0 |
| 61–90 | 30 | 16.0 |
| 91–120 | 31 | 15.9 |
| 121–150 | 22 | 17.6 |
| 151–180 | 22 | 17.6 |

**Table 2. Time per frame and speed-up results for animation Test 4. The average speed-up of the method was 16.8**

| $\frac{A_d}{A}$ | time per frame (s) | Speed-up |
|---|---|---|
| 0.3 | 25.8 | 17.8 |
| 0.2 | 23 | 19.9 |
| 0.08 | 20 | 23.0 |
| 0.03 | 17.6 | 25.9 |

**Table 3. Time per frame and speed-up for the flying plane animation of Test 2 for different lengths of motion path**

**Test 3: Several Simultaneous Motions.** In this environment several cubes are animated, each one following a different motion path. Figure 11 shows four frames of one of the sequences. Maintaining the total number of polygons constant, we vary the number of animated cubes, so the number of dynamic polygons changes. All animations were processed with 60 frames. We can see (Table 4) that the speed-up decreases as more objects are animated, but we also note that this decrease is lower than the proportion of the ratio $\frac{n_d}{n}$ increased (Figure 12), which means that although the 'complexity' of the animation is increased the method still achieves efficiency.

## Summary of Results

The key to the efficiency of this new method is that static surface intersections and their corresponding interaction are reused for different frames. Consider the different possible interactions between static surfaces: static–static and static–dynamic interactions. If a static–static interaction occurs, then, for the whole interval of the animation, $k$ different simulating paths of light, one for each frame, are updated using the same intersections. Also, if a static–dynamic interaction

| $n_s$ | $n_d$ | $\frac{n_d}{n}$ | Time pr frame (s) | Speed-up |
|---|---|---|---|---|
| 84 | 6 | 0.06 | 10 | 19.3 |
| 72 | 18 | 0.20 | 16 | 12.4 |
| 60 | 30 | 0.33 | 23 | 9.0 |
| 42 | 48 | 0.53 | 29 | 7.0 |
| 30 | 60 | 0.66 | 32 | 6.4 |

**Table 4. Speed-up results for animation Test 3 when we vary the number of dynamic objects**

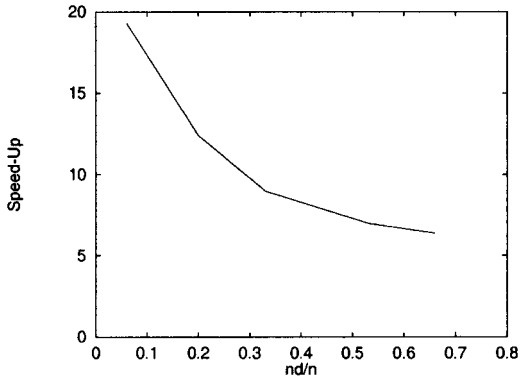


*Figure 12. Speed-up vs. ratio of the number of dynamic polygons to the total number of polygons. The speed-up decreases as more objects are animated.*

occurs in the visibility list, the static surface intersection is used to update many different simulating paths of light for different instances of the dynamic object. From our tests we can conclude about the efficiency of the proposed method that:

- It increases as more frames are processed together, but the efficiency is bounded, meaning that after gathering a number of frames it remains almost constant.
- It increases if the size of the motion volume of the dynamic object is reduced while computing the same number of frames. When the animation takes place in a reduced space with respect to the whole scene's space, the intersection processing of the dynamic part is optimal. This can also be observed from the last term of equation (5), indicating that the ratio between the area of the bounding motion volume and the area of the bounding scene $\frac{A_d}{A}$ is proportional to the cost.
- It increases if the ratio between static and dynamic surfaces grows (for instance, by adding more static surfaces). In this case more static–static interactions are computed and reused for many frames.
- It decreases if the number of animated objects following different motion paths is increased. In this case the ratio of static to dynamic surfaces decreases and also more motion volumes need to be tested for intersection. We can consider this case as an increase of the complexity of the animation.

# Final Discussion

In the context of related work on animated radiosity environments we compare fundamental features of previous techniques with our approach, pointing out the benefits obtained:

- *Monte Carlo simulation*: Our approach is completely based on a Monte Carlo simulation, not used previously for animated environments. In this way, we can take advantages of such kinds of simulation: Monte Carlo radiosity provides a reliable form factor computation, while no explicit storage of the form factor is needed. Thus, as the whole memory storage is optimized, we have the possibility to deal with more complex scenarios than deterministic techniques.
- *Visibility change management*: No explicit visibility structure is used to manage the visibility changes problem, a very important aspect of animated global illumination methods. The main important structures used by previous techniques for this purpose are summarized in the 'Background' section above. In our approach the problem is handled by a global Monte Carlo technique, from which we can encode the visibility information for different instants of time in a relatively simple way. Therefore, we optimize the visibility changes computation both in memory and time.
- *View dependency*: No restriction to the point of view is imposed to our method. In Reference 8 animations are conceived from a fixed point of view. In other approaches, as in References 10 and 12, changing the camera path completely leads to some recomputations of illumination. The solution obtained with our method is completely view independent.
- *Speed-up*: A relatively high speed-up is obtained with our method. Depending on animation parameters, such as the number of dynamic objects, the size of the motion volume of the moving objects or the ratio between static and dynamic objects, we obtained speed-ups in the range of 7–25 without any loss of quality in the light simulation.

# Other Issues and Future Work

## Motion Blur Effects

Motion blur effects are frequently used in animations with the goal of increasing realism. To keep a realistic simulation of such effect, a physically based lighting method must be improved to compute all light energy interactions between surfaces in the time interval the camera shutter is open. Although several motion blur algorithms have been proposed for different rendering methods, very few consider the computation of all

diffuse interreflections in such a time interval. Two different approaches can be used to generate motion blur effects within the MFLM. In Besuievsky and Pueyo,[34] an accurate method that solves the illumination equation over time is described. An alternative is proposed in Besuievsky and Pueyo,[31] considering the use of an interpolated-based approach, to obtain an approximated solution of the motion blur effect. First, we decide on a number of samples to take between successive frames, then interpolated solutions for each sample are obtained from the full solutions computed at the frames by interpolation. Finally, each solution is projected and combined using an accumulation buffer approach in a similar way as in Haeberky and Akeley.[35]

## Adaptive Meshing

An adaptive strategy to deal with the meshing of surfaces would improve our method. In general, this is a critical aspect in all Monte Carlo radiosity algorithms. We use a simple regular mesh subdivision. If a very fine mesh is used to obtain an accurate illumination computation, many lines are needed in order to avoid noise artifacts. When dealing with dynamic environments, in addition, variation of the meshing according to the amount of energy of the surface exchange may be considered. The combination of our method with a hierarchical radiosity approach, like that described in Bekaert *et al*.,[27] is considered for future work.

## Radiosity Interpolation

An interpolation-based improvement can be used to accelerate the method. The fact that illumination changes in animations are smooth has been used to exploit coherence in animated environments. Nimeroff *et al*.[10] introduced interpolation of indirect illumination in their framework for animated environments. We propose the use of interpolation of radiosities at object space.[32] The MFLM is performed in the same way as in the original proposition but with a lower number of frames. Intermediate radiosity solution frames are obtained by interpolation of exactly computed ones. In this way both time processing and memory requirements are improved. The cost of processing decreases because fewer objects are intersected and less memory storage is required as only the fully computed frames are recorded.

## Distribution Processing and Progressive Refinement

The MFLM is suited to distribution among different processors to accelerate the total processing time of the method. Following the splitting strategy described in the section on 'Implementation and Usability' above we can distribute each resulting sub-animation in a straightforward way.[32] No communication is needed between the processes.

Another possibility for distribution is using the idea of merging solutions presented in 'Sbert' *et al*.[36] In Monte Carlo algorithms the variances are inversely proportional to the number of rays cast. Therefore, instead of casting all $N_T$ rays at once, to obtain a solution, we can divide the problem into smaller ones and obtain a solution incrementally with $n$ results of $\frac{N_T}{n}$ rays. This approach also gives us the possibility of refinning progressively the solution. In this distributed approach all $n$ solutions must be computed separately and combined later to obtain the final solution.

#### ACKNOWLEDGEMENTS

The authors wish to thank all onu-office members for much help in the development of this work. Thanks also to Mateu Sbert for collaborating and discussing results at the beginning of this work, which has been partly supported by project TIC 98-0586-03-02 of the CICYT, and grant 1999SGR-00162 of the DGR-Generalitat de Catalunya.

# References

1. Goral CM, Torrance KE, Greenberg DP, Battaile B. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (ACM SIGGRAPH'84 Proceedings)*, Vol. 18, July 1984; pp 212–222.
2. Chen SE. Incremental radiosity: an extension of progressive radiosity to an interactive image synthesis system. In *Computer Graphics (ACM SIGGRAPH'90 Proceedings)*, Vol. 24, August 1990; pp 135–144.
3. George DW, Sillion FX, Greenberg DP. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications* 1990; **10**(4): 26–34.
4. Müller S, Schoeffel F. Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor-list. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June 1994; pp 325–342.
5. Shaw E. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum* 1997; **16**(2): 107–118.
6. Drettakis G, Sillion FX. Interactive update of global illumination using a line-space hierarchy. In *Computer*

*Graphics (ACM SIGGRAPH'97 Proceedings)*, Vol. 31, 1997; pp 57–64.
7. Schoeffel F, Pomi A. Reducing memory requirements for interactive radiosity using movement prediction. In *Rendering Techniques'99 (Proceedings of the Tenth Eurographics Workshop on Rendering)*, Lischinski D, Larson G (eds). Springer: New York, 1999; pp 225–234.
8. Baum DR, Wallace JR, Cohen MF, Greenberg DP. The back-buffer algorithm: an extension of the radiosity method to dynamic environments. *Visual Computer* 1986; **2**(5): 298–306.
9. Forsyth DA, Yang C, Teo K. Efficient radiosity in dynamic environments. In *Fifth Eurographics Workshop on Rendering*, Darmstadt, Germany, June 1994; pp 313–323.
10. Nimeroff J, Dorsey J, Rushmeier H. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics* 1996; **2**(4): 283–298.
11. Damez C, Sillion F. Space-time hierarchical radiosity. In *Rendering Techniques'99 (Proceedings of the Tenth Eurographics Workshop on Rendering)*, Lischinski D, Larson G (eds). Springer: New York, 1999; pp 235–246.
12. Martín I. *Temporal Coherence in Animation of Global Illumination Environments*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, 2000.
13. Beseuievsky G, Sbert M. The Multi-Frame Lighting Method: a Monte Carlo based solution for radiosity in dynamic environments. In *Rendering Techniques'96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, Pueyo X, Schröeder P (eds). Springer: New York, 1996; pp 185–194.
14. Haines EA, Wallace JR. Shaft culling for efficient ray-traced radiosity. In *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, Brunet P, Jansen FW (eds). Springer: New York, 1994.
15. Orti R, Riviere S, Durand F, Puech C. Radiosity for dynamic scenes in Flatland with the visibility complex. *Computer Graphics Forum (Eurographics'96)* 1996; **15**(3): C237–C248.
16. Durand F, Drettakis G, Puech C. The 3D visibility complex: a new approach to the problems of accurate visibility. In *Rendering Techniques'96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, Pueyo X, Schröeder P(eds). Springer: New York, 1996; pp. 245–268.
17. Durand F, Drettakis G, Puech C. The visibility skeleton: a powerful and efficient multi purpose global visibility tool. In *Computer Graphics (ACM SIGGRAPH'97 Proceedings)*, Vol. 31, 1997; pp 89–100.
18. Buckalew C, Fussell D. Illumination networks: fast realistic rendering with general reflectance functions. In *Computer Graphics (ACM SIGGRAPH'89 Proceedings)*, Vol. 23, July 1989; pp 89–98.
19. Sbert M. An integral geometry based method for fast form factor computation. *Computer Graphics Forum (Eurographics'93)* 1993; **12**(3): C409–C420.
20. Neumann L. Monte Carlo radiosity. *Computing* 1995; **55**(1): 23–42.
21. Pellegrini M. Monte Carlo approximation of from factors with error bounded a priori. In *Eleventh ACM Symposium on Computational Geometry*, Vancouver, BC, June 1995; pp 287–296.
22. Shirley P. Radiosity via ray tracing. In *Graphics Gems II*, Arvo J (ed.). Academic Press: Boston, MA, 1991; pp 306–310.
23. Pattanaik SN, Mudur SP. Computation of global illumination by Monte Carlo simulation of the particle model of light. In *Third Eurographics Workshop on Rendering*, Bristol, UK, May 1992; pp 71–83.
24. Feda M, Purgathofer W. Progressive ray refinement for Monte Carlo radiosity. In *Fourth Eurographics Workshop on Rendering*, Paris, France, June 1993; pp 15–26.
25. Neumann L, Neumann A, Bekaert P. Radiosity with well distributed ray sets. *Computer Graphics Forum (Eurographics'97)* 1997; **16**(3): C261–C269.
26. Sbert M, Pueyo X, Neumann L, Purgathofer W. Global multipath Monte Carlo algorithms for radiosity. *Visual Computer* 1996; **12**(2): 47–61.
27. Bekaert P, Neumann L, Neumann A, Sbert M, Willems YD. Hierarchical Monte Carlo radiosity. In *Rendering Techniques'98 (Proceedings of Eurographics Rendering Workshop'98)*, Drettakis G, Max N (eds). Springer: New York, 1998; pp 259–268.
28. Szirmay-Kalos L, Purgathofer W. Global ray-bundle tracing with hardware acceleration. In *Rendering Techniques'98 (Proceedings of Eurographics Rendering Workshop'98)*, Drettakis G, Max N (eds). Springer: New York, 1998; pp 247–258.
29. Santaló LA. *Integral Geometry and Geometric Probability*. Addison-Wesley: New York, 1976.
30. Besuievsky G. Making global Monte Carlo methods useful: an adaptive approach for radiosity. In *Actas VII Congreso Espanol de Informatica Grafica (CEIG '97)*, Barcelona, Spain, June 1997.
31. Besiuevsky G, Pueyo X. A dynamic light source algorithm for radiosity environments. In *Computer Animation and Simulation'98*, Arnaldi B, Hegron G (eds). Springer: New York, 1998.
32. Besuievsky G. *A Monte Carlo Approach for Animated Radiosity Environments*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, 2000.
33. Martín I, Pérez F, Pueyo X. The SIR rendering architecture. *Computers & Graphics* 1998; **22**(5): 601–609.
34. Besuievsky G, Pueyo X. A motion blur method for animated radiosity environments. In *WSCG '98 (Sixth European Conference in Central Europe on Computer Graphics and Visualization)*, Skala V (ed.). Plzen, Czech Republic, 1998. University of West Bohemia.
35. Haeberky P, Akeley K. The accumulation buffer. In *Computer Graphics (ACM SIGGRAPH'90 Proceedings)*, Vol. 23, 1990; 309–318.
36. Sbert M, Pérez F, Pueyo X. Global Monte Carlo: a progressive solution. In *Rendering Techniques'95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, Hanrahan PM, Purgathofer W (eds). Springer: New York, 1995; pp 231–239.