# Simultaneous Progressive Refinement in Dynamic Environments

**Jens-Uwe Hahn**

**ISSN 0946-3852**
**WSI-98-3**

Wilhelm-Schickard-Institut für Informatik
Graphisch-Interaktive Systeme
Auf der Morgenstelle 10/C9
D-72076 Tübingen
Tel.: +49 7071 29-75462
Fax: +49 7071 29-5466

email: jhahn@gris.uni-tuebingen.de

anonymous but otherwise identical version submitted to SIGGRAPH98 (January 1998)

## Abstract

This paper presents a new method for efficient radiosity calculation in dynamic environments. The method is intended for animation sequences in which the motion and mutation of the objects is known in advance. The algorithm is based on Wallace's progressive refinement algorithm. Progressive refinement iterations are performed simultaneously for all frames of the animation. Being precise, we give a proof of the convergence of our algorithm to the correct solution. Using our technique, a smooth non flickering illumination is guaranteed during the animation sequence.

**CR Categories:** I.3.7 [Three-Dimensional Graphics and Realism]: Radiosity; I.3.7 [Three-Dimensional Graphics and Realism]: Animation

**Keywords:** Global illumination, Radiosity, Dynamic environment, Progressive Refinement

## 1 Introduction

Physically correct illumination in three-dimensional scenes is of essential importance for realistic image synthesis. Radiosity [13, 7, 16] is an efficient and well known method to obtain solutions in diffuse static environments. Realistic animations and interactive applications, however, require illumination techniques in dynamic environments, i.e. environments where the appearance, location or shape of objects may change.

The Radiosity method calculates an equilibrium of energy transfer between the surfaces in the scene. This equilibrium is based on visibility and form factors [9] between pairs of surfaces. All form factors may possibly be affected by any change of the scene geometry. The visibility of two surfaces may be blocked by a moving object, or surfaces may become visible when the occluding object is removed. Also the radiosity values of all surfaces may change due to direct or indirect effects of the geometrical change.

In practice however, particular changes in the scene will result in only inconsiderable changes of the radiosity solution in distant parts of the scene. If someone for example moves a book on the desk in an office room, the illumination of the floor, the ceiling and most of the walls will not change noticably. Only a few form factors and only the visibility between a few surfaces will change.

Therefore a great amount of computation can be saved by the use of algorithms which avoid the recomputation of those visibilities and form factors which do not change during particular intervals of time.

We may distinguish two types of algorithms for radiosity in dynamic environments: Algorithms that need to know all changes in the scene already at the beginning of the radiosity calculation, and algorithms that start their calculation with an initial scene and get the solutions for subsequent frames by updating the solution of the previous frame. Algorithms that need to know all changes in advance are suitable for the calculation of video animations, while interactive applications require algorithms that are able to update an existing solution after modifications of the environment. In general, the latter algorithms are also suitable to perform the calculations of video animations. But we will see in section 3 that update algorithms that are based on progressive refinement lead to artifacts unless a very high amount of computational effort is used.

In this paper we present a method that calculates a radiosity solution in dynamic environments simultaneously for all frames. Therefore the changes in the scene must be known in advance. The method is based on Wallace's progressive refinement algorithm [18] and can deal with changes of geometry, materials and light emission.

In section 2 we describe previous work related to radiosity in dynamic environments. In section 3 problems particular about progressive refinement for video animations are discussed. And finally our algorithm is described in section 4 followed by some results presented in section 5.

## 2 Previous Work

The first algorithm dealing with radiosity in dynamic environments was presented by Baum et. al. already in 1986 [1], just two years after the first paper on radiosity [13] has been published. This algorithm is based on a full matrix solution using hemicube form factors [6]. Swept volumes restricting the part of space affected by moving objects are used to predict visibility changes between the different frames. A first pass of the algorithm calculates and stores hemicubes around all static patches of the scene. All static patches and the volume swepts of all dynamic patches are projected onto these hemicubes, which are used in a second pass to calculate a radiosity solution for each frame of the animation. Obviously, a very high amount of memory and computation power is required by this algorithm which is therefore not suitable for complex environments.

Other approaches are based on the progressive refinement method introduced in [5]. In [4] and [12] the authors present methods which incrementally update the radiosity solution after modifications of the scene. Corrections are propagated through the environment by some additional progressive refinement iterations. In [15] we find an algorithm which uses a Shadow-Form-Factor-List. During initial progressive refinement calculation, the form factors and shadowing objects are stored in this list for every progressive refinement step. After modifications of the scene, the previously performed progressive refinement iterations can be matched to the new environment with the use of this list. If a surface which has not previously been selected as a secondary light source becomes highly illuminated due to modifications of the scene, some additional progressive refinement steps need to be performed. If the modifications of the scene are known in advance, those additional steps can be avoided by manipulating the progressive refinement sequence of the initial calculation by hand.

In [10] a solution based on hierarchical radiosity is obtained by updating links between surfaces dependent on the motion of objects. In [8] the authors present an update algorithm in the context of clustering for hierarchical radiosity using a line-space hierarchy. Links between hierarchical elements are updated when the scene geometry changes, and additional bounces may need to be performed. This algorithm provides a control mechanism allowing for the regulation between image quality and frame rate.

All the progressive refinement algorithms cited above are designed to work in interactive applications. But for video animations they have to deal with problems described in the following section.

A method that is based on global Monte Carlo radiosity can be found in [2]. Intersections of global lines [2] with the environment are calculated in this approach with a scene that is merged together from all frames. This algorithm suffers from the fact that global Monte Carlo radiosity needs a high amount of computation power.

## 3 Progressive Refinement in Animations

The progressive refinement algorithm ensures that in every iteration the most powerful surface will shoot. Thus convergence of the radiosity solution is very fast for the first few iterations. But as the solution approaches the exact solution the rate of convergence is decreasing rapidly. Good visual results are usually obtained already after a few iterations of progressive refinement, even though the absolute error is still very high.

The progressive refinement method is suitable to achieve good results for photo-realistic image synthesis in complex environments. Only a few surfaces need to propagate their energy through

the scene; thus only a few columns of the form factor matrix have to be calculated. But there will remain a relatively high absolute error.

In order to get a solution of a modified scene by updating the original solution, it might not be sufficient to do updates of the previously performed iterations. Surfaces which have not been propagating their energy through the scene in the previous calculation might become highly illuminated in the modified environment. Important global illumination effects may be caused by indirect light leaving those surfaces. Therefore we need to perform some additional progressive refinement steps.

But on the other hand, if the sequence of progressive refinement steps is not the same in all frames, the remaining error will be distributed differently in the scene in subsequent frames. This will result in a flickering animation, which is surely no problem for interactive applications, but unacceptable for video animations.

If all changes in the scene are known in advance, a sequence of progressive refinement steps which is suitable for all frames can be found; and all frames can be calculated with the same sequence resulting in a smooth, non flickering animation. An algorithm to achieve this will be described in the following.

# 4 Simultaneous Progressive Refinement

Next we will describe our algorithm for simultaneous progressive refinement and give a proof of the convergence of this algorithm.

## 4.1 The Algorithm

The algorithm is based on a scene description where every object in the scene is aware of its motion and mutation during the animation sequence. The progressive refinement procedure is performed simultaneously for all frames.

Let $F_i,\ i = 1, \ldots, n$ denote the faces in the scene and $P_{i,f}$ the unshot power of face $F_i$ in frame $f,\ f = 1, \ldots, m$. Face $F_i$ is said to be *a most powerful face*, if

$$\max_{f=1,\ldots,m} P_{i,f} \geq \max_{\substack{f=1,\ldots,m \\ j=1,\ldots,n}} P_{j,f}. \tag{1}$$

For a progressive refinement step a most powerful face $L$ is selected with the use of the above definition. This face is used to propagate its unshot radiosity through the environment simultaneously for all frames.

If face $L$ is a dynamic face (i.e the face changes position or shape during the animation sequence), visibility detections and form factor calculations to receiving faces have to be done for every frame. However, if face $L$ is static (i.e. the face does not change position or shape during the animation sequence), form factors to static receiver faces remain constant during the complete animation sequence. Also visibility detection to static receiver faces can be done simultaneously for all frames by tracing rays starting at receiver vertices and ending at emitter samples through the environment. Starting points and end points of these rays remain constant for all frames.

Our progressive refinement algorithm can be stated in C-like pseudo-code as shown in figure 1:
The radiosity propagation in the first case where $L$ is dynamic can be formulated as described in figure 2:
For the second case where $L$ is static, the radiosity propagation is performed as shown in figure 3:

In the case where the emitting face $L$ and the receiving face $F$ are both static, rays $R : v \rightarrow s$ have to be traced through the environment to detect visibility simultaneously for all frames.

```
while (more iterations to be performed) {
  select a most powerful face L
  if (L is dynamic)
    propagate radiosity separately for each frame
  else              // L is static
    propagate radiosity simultaneously for all frames
}
```

Figure 1: Simultaneous Progressive Refinement

```
for (all frames f, faces F, vertices v of F and samples s of L) {
  trace ray v→s through scene
  if (visible) {
    calculate form factor
    transfer radiosity from s to v
  }
}
```

Figure 2: Radiosity Propagation for Dynamic Light Sources

For this purpose every ray $R : v \rightarrow s$ is equipped with a visibility mask containing a flag for each frame. These flags are used to mark visibility or occlusion for the particular frames. Initially all frames are marked to have visibility. As ray $R$ is traced through the environment, intersection tests with static objects have to be done only once. If ray $R$ hits a static object, all frames can be marked occluded. Intersection tests with dynamic objects have to be done for those frames which are not already marked occluded. The tracing process can be aborted as soon as all frames are marked occluded.

In the example shown in figure 4 a ray $R$ starting at a static receiver vertex $v$ and ending at a static emitter sample $s$ is traced through a dynamic environment. The animation sequence consists of five frames. The visibility mask of the ray is initialized with 1 for each frame (a) meaning that we have visibility in all frames. The intersection test with the static object $O_1$ has to be performed only once leading to the result that the object does not occlude the ray in any frame (b). The intersection test with the rectangular dynamic object $O_2$ gives occlusion in frame two and three which is marked in the visibility mask of the ray by setting the flags of frame two and three to 0 (c). The intersection test with the oval dynamic object $O_3$ can ignore frame two and three since these frames are already marked occluded. Frame four is marked occluded due to this object (d). The visibility mask at the end of the tracing procedure (e) shows that we have visibility in frame one and five and occlusion in the frames two, three and four.

Then the form factor is calculated and the radiosity of the emitter is transfered to the receiver for frame one and five.

## 4.2 Convergence of Simultaneous Progressive Refinement

In the first subsection we will briefly present the concept of relaxation since it will be used in the following to prove convergence of the algorithm. The proof is similar to the one in [14] given for progressive refinement in static environments. For more detailed information about relaxation see also [3] and [11].

### 4.2.1 Relaxation

Let us consider the linear system

$$M x = b$$

where $M$ is an $n \times n$ matrix and $x, b \in \mathbf{R}^n$. To solve this linear system for $x$ we want to use an iterative method, starting with an approximate solution $x^{(0)}$. For the approximate solution $x^{(k)} =$

```
for (all faces F) {
  if (F is dynamic) {
    for (all frames f, all vertices v of F and all samples s of L) {
      trace ray v→s through scene
      if (visible) {
        calculate form factor
        transfer radiosity from s to v
      }
    }
  } else { // F is static
    for (all vertices v of F and all samples s of L) {
      trace ray v→s through scene
      if (visible in at least one frame) {
        calculate form factor
        for (all frames in which we have visibility)
            transfer radiosity from s to v
      }
    }
  }
}
```

Figure 3: Radiosity Propagation for Static Light Sources

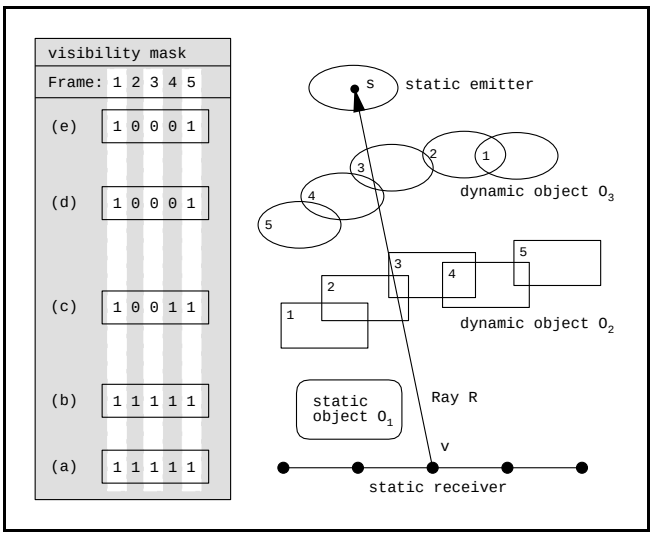


Figure 4: Simultaneous Ray Tracing using Visibility Masks. (a)–(e): progression of the visibility status for ray R

$(x_1^{(k)}, \ldots, x_n^{(k)})$ obtained after the $k^{th}$ step, we define the $k^{th}$ error $e^{(k)}$ and the $k^{th}$ residual $r^{(k)}$ as

$$e^{(k)} := x - x^{(k)} \quad and \quad r^{(k)} := b - Mx^{(k)} = Me^{(k)}.$$

If the residuals $r^{(k)} = (r_1^{(k)}, \ldots, r_n^{(k)})$ converge to zero, then the $x^{(k)}$ converge to the correct solution. Relaxation is a method that attempts to achieve this. The idea is to *relax* one variable $x_i^{(k)}$ in every iteration, i.e. change this variable in such a way that $r_i^{(k+1)} = 0$. Of course other $r_j^{(k)}$ may increase thereby. But if the variables to be relaxed are chosen properly, convergence can be achieved (see below and 4.2.2).
If the variable $x_i^{(k)}$ is to be relaxed, we need to set

$$x_i^{(k+1)} := \frac{b_i - \sum_{j \neq i} M_{ij} x_j^{(k)}}{M_{ii}}.$$

Using the definition of $r^{(k)}$, i.e.

$$r_i^{(k)} = b_i - \sum_j M_{ij} x_j^{(k)}, \tag{2}$$

we obtain

$$x_i^{(k+1)} = x_i^{(k)} + \frac{r_i^{(k)}}{M_{ii}}.$$

Applying equation (2) to $r_j^{(k)}$ and $r_j^{(k+1)}$, we get for the new residual

$$r_j^{(k+1)} = \begin{cases} r_j^{(k)} - \frac{M_{ji}}{M_{ii}} r_i^{(k)} & \text{for } j \neq i, \\ 0 & \text{for } j = i. \end{cases} \tag{3}$$

If we relax the $x_i^{(k)}$ in order, we get the well known Gauss-Seidel iteration algorithm, which converges for diagonally dominant matrices. If in every iteration step that variable $x_i^{(k)}$ with the greatest residual $r_i^{(k)}$ is relaxed, then the method is called *Southwell Relaxation*. In [14] the authors prove that this method converges for strictly column diagonally dominant matrices by showing that the total residual decreases by some constant factor in every iteration step.

### 4.2.2 Proof of Convergence

The radiosity solution in a static environment is obtained by solving the system of linear equations given by

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ji} \frac{A_j}{A_i}, \quad i = 1, \ldots, n$$

where the scene consists of $n$ patches and

| | | |
|---|---|---|
| $B_i$ | = | the radiosity of patch $i$ |
| $E_i$ | = | the emission of patch $i$ |
| $A_i$ | = | the area of patch $i$ |
| $\rho_i$ | = | the reflectivity of patch $i$ |
| $F_{ij}$ | = | the form factor from patch $i$ to patch $j$. |

With the use of the reciprocity relationship $F_{ji}A_j = F_{ij}A_i$, we obtain

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}, \quad i = 1, \ldots, n. \tag{4}$$

Instead of solving system (4) we can, again with the use of the reciprocity relationship, equivalently solve the system

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j A_j F_{ji}, \quad i = 1, \ldots, n \tag{5}$$

or in matrix form

$$M\ B = \tilde{E}$$

where

$$M = \begin{bmatrix} A_1 - \rho_1 A_1 F_{11} & \dots & -\rho_1 A_n F_{n1} \\ -\rho_2 A_1 F_{12} & A_2 - \rho_2 A_2 F_{22} & \\ \vdots & & \vdots \\ -\rho_n A_1 F_{1n} & \dots & A_n - \rho_n A_n F_{nn} \end{bmatrix},$$

$$B = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} \text{ and } \tilde{E} = \begin{bmatrix} E_1 A_1 \\ E_2 A_2 \\ \vdots \\ E_n A_n \end{bmatrix},$$

which is a strictly column diagonally dominant system since $\sum_{j=1,\dots,n} F_{ij} < 1$ and $\rho_i < 1$ for all $i = 1, \dots, n$.

In [14] the authors show that performing a progressive refinement iteration to equation (4) where the most powerful patch is shooting is equivalent to performing a Southwell Relaxation step to equation (5) where the variable with the greatest residuum is relaxed and the output is the variables added to their residuals. The convergence of the progressive refinement method for the radiosity equation (4) is therefore equivalent to the convergence of Southwell Relaxation for equation (5).

To obtain a radiosity solution for a dynamic environment, we have to solve equation (4) for each frame. For an animation sequence consisting of $m$ frames, we have to solve

$$B_i^f = E_i^f + \rho_i^f \sum_{j=1}^{n} B_j^f F_{ij}^f, \quad i = 1, \dots, n, \ f = 1, \dots, m \qquad (6)$$

or the equivalent system in matrix form

$$\begin{bmatrix} M^1 & 0 & \dots & 0 \\ 0 & M^2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & M^m \end{bmatrix} \begin{bmatrix} B^1 \\ B^2 \\ \vdots \\ B^m \end{bmatrix} = \begin{bmatrix} \tilde{E}^1 \\ \tilde{E}^2 \\ \vdots \\ \tilde{E}^m \end{bmatrix}, \qquad (7)$$

which is still a strictly column diagonally dominant system. The matrix is a block matrix consisting of one block for each frame of the animation. The values in the particular frames are denoted by the superscript index.

Performing one simultaneous progressive refinement step as described above is now equivalent to performing $m$ relaxation steps to equation (7). These $m$ relaxation steps relax exactly one variable in every block of the matrix and are therefore independent of each other in the sense that the residuals of a relaxed variable will not become nonzero due to the other relaxation steps. That is, $m$ variables are relaxed simultaneously in one simultaneous progressive refinement step.

In the following, we will show that the total residual decreases by some constant factor in each simultaneous progressive refinement step. For convenience we use the $l_1$ norm:

$$\|r\| := \sum_i |r_i|.$$

Let us suppose, patch $F_i$ is propagating its unshot radiosity through the scene in the $(k+1)^{th}$ iteration step. Since equation (3) applies to every block of our matrix, we have

$$\begin{aligned} \|r^{(k+1)}\| &= \|r^{(k)}\| - \sum_{f=1}^{m} |r_i^{f\ (k)}| + \sum_{f=1}^{m} \sum_{j \neq i} \left| \frac{M_{ji}^f}{M_{ii}^f} \, r_i^{f\ (k)} \right| \\ &= \|r^{(k)}\| - \sum_{f=1}^{m} |r_i^{f\ (k)}| + \sum_{f=1}^{m} |r_i^{f\ (k)}| \sum_{j \neq i} \left| \frac{M_{ji}^f}{M_{ii}^f} \right|. \end{aligned}$$

Since our matrix is strictly column diagonally dominant, we get

$$p := \max_{i,f} \sum_{j \neq i} \left| \frac{M_{ji}^f}{M_{ii}^f} \right| < 1$$

and therefore

$$\|r^{(k+1)}\| \leq \|r^{(k)}\| - (1-p) \sum_{f=1}^{m} |r_i^{f\ (k)}|. \qquad (8)$$

In our algorithm we always choose a most powerful face (see (1)) as shooting patch. With $P_j^f = B_j^f A_j^f$, we therefore achieve that the shooting patch $F_i$ is chosen such, that

$$|r_i^{\tilde{f}\ (k)}| \geq \max_{j,f} |r_j^{f\ (k)}| \quad \text{for some } \tilde{f} \in \{1, \dots, m\}$$

and thus

$$\sum_{f=1}^{m} |r_i^{f\ (k)}| \geq |r_i^{\tilde{f}\ (k)}| \geq \frac{\|r^{(k)}\|}{n\ m}.$$

With the use of (8), we now get

$$\|r^{(k+1)}\| \leq \|r^{(k)}\| - \frac{(1-p)}{n\ m} \|r^{(k)}\| = q\ \|r^{(k)}\|,$$

where

$$q = 1 - \frac{1-p}{n\ m} < 1.$$

It follows that

$$\|r^{(k)}\| \leq q^k\ \|r^{(0)}\|,$$

which converges to zero as $k$ goes to infinity. □

## 5 Implementation and Results

In this section we present some results to evaluate the efficiency of our algorithm.

The new algorithm has been integrated in the object oriented global illumination system RadioLab [17]. The implementation has been done in C++. All results are computed on a Silicon Graphics $O_2$ with an 180 MHz R5000 processor.

The example scene shown in figure 5 contains 20,139 patches. This scene is composed of an office room, which is brightly illuminated by three light sources, and a living room, which is only illuminated by a television screen. During the animation sequence, a door between these two rooms is being opened and a movie on the television screen is shown. The illumination effects of the opening door as well as the changing color in the room due to the color bleeding caused by the changing images of the movie on the screen can be seen clearly in figure 5.

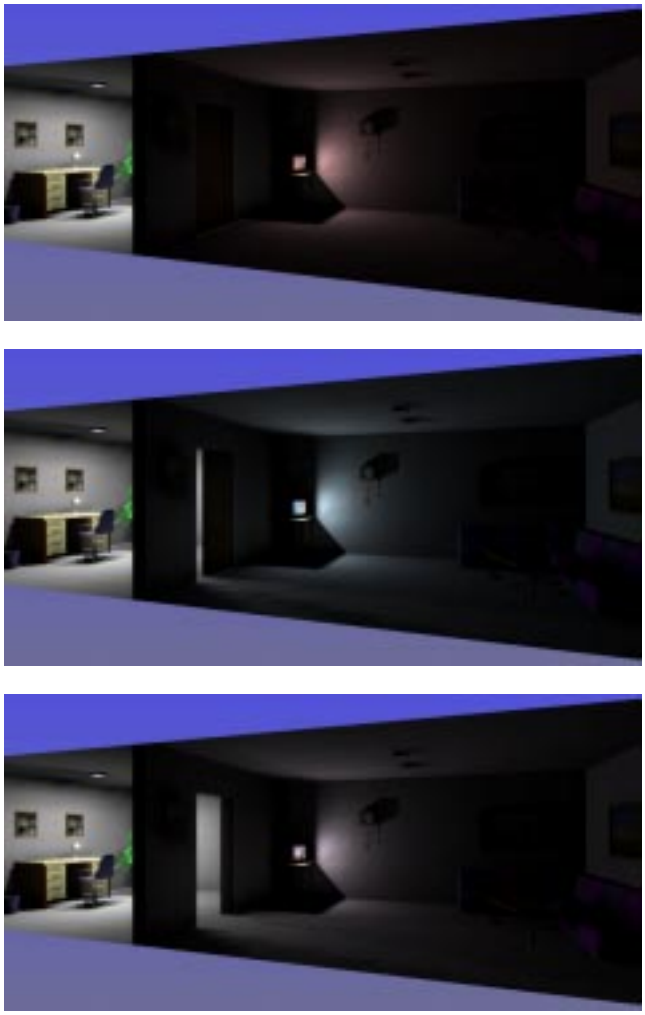

Figure 5: Example scene

We calculated an animation sequence consisting of 40 frames. For acceleration of ray tracing, which is used for visibility detection, a 3D grid with a resolution of 40 times 16 times 11 voxels was used. Highly occupied voxels were subdivided recursively.

The simultaneous progressive refinement calculation resulting in the images shown in figure 5 took 224 seconds for all 40 frames, whereat 4,309,573 intersection calculations had to be performed. For the same iterations for one single frame, 95 seconds and 3,475,145 intersection calculations were needed. The average time needed for an additional frame therefore is 3.3 seconds or 3.5% of the time needed for a single frame. The average number of intersection calculations needed for an additional frame is 21,396 or 0.6% of those needed for a single frame.

## 6 Conclusion and Future Work

We have presented a new method for radiosity calculation in dynamic environments based on Wallace's progressive refinement algorithm. The calculation is done simultaneously for all frames. The effort for visibility detection and form factor calculation between static surfaces has to be taken only once for the complete animation sequence. The fact that the sequence of progressive refinement iterations is the same for all frames of the animation ensures a smooth non flickering illumination of the environment. The method seems to be a practical technique for photo-realistic animations. But, since the changes of the scene have to be known in advance, it is not usable for interactive applications.

Our future research will include improvements of performance and storage costs of the algorithm and the quality of the resulting images. The use of space-time coherence of dynamic objects should result in a significant improvement of the performance of the algorithm. In order to reduce storage costs, a proper dynamic representation of the radiosity function on the surfaces has to be developed, which is an important issue for longer video animations in complex environments. And finally dynamic meshing strategies are needed as discontinuities can change during the animation sequence.

## References


[1] Daniel R. Baum, John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments. *The Visual Computer*, 2(5):298–306, September 1986.

[2] Gonzalo Besuievsky and Mateu Sbert. 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)*, pages 185–194, New York, NY, 1996. Springer-Verlag/Wien.

[3] R. Burlisch and J. Stoer. *Introduction to Numerical Analysis*. Springer-Verlag, Berlin, Germany, 1972.

[4] Shenchang E. Chen. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 135–144, August 1990.

[5] Michael Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.

[6] Michael Cohen and Donald P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, volume 19, pages 31–40, August 1985.

[7] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993.

[8] George Drettakis and François Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics Proceedings, Annual Conference Series:* SIGGRAPH '97 (Los Angeles, CA), pages 57–64. ACM SIGGRAPH, New York, August 1997.

[9] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1990.

[10] David A. Forsyth, Chien Yang, and Kim Teo. Efficient Radiosity in Dynamic Environments. In *Fifth Eurographics Workshop on Rendering*, pages 313–323, Darmstadt, Germany, June 1994.

[11] N. Gastinel. *Linear Numerical Analysis*. Academic Press, Boston, MA, 1970.

[12] David W. George, Francois X. Sillion, and Donald P. Greenberg. Radiosity Redistribution for Dynamic Environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990.

[13] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, July 1984.

[14] Steven J. Gortler, Michael F. Cohen, and Phillipp Slusallek. Radiosity and Relaxation Methods: Progressive Refinement is Southwell Relaxation. Technical Report CS-TR-408-93, Department of Computer Science, Princeton University, Princeton, NJ, February 1993.

[15] Stefan Muller and Frank Schoffel. Fast Radiosity Repropagation for Interactive Virtual Environments Using a Shadow-Form-Factor-List. In *Fifth Eurographics Workshop on Rendering*, pages 325–342, Darmstadt, Germany, June 1994.

[16] Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994.

[17] R. Sonntag. *RadioLab: An object oriented global illumination system*. Dissertation, Universität Tübingen, Tübingen, Germany, 1998. Forthcoming.

[18] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A Ray Tracing Algorithm for Progressive Radiosity. In *Computer Graphics (ACM SIGGRAPH '89 Proceedings)*, volume 23, pages 315–324, July 1989.