

Hierarchical Improvement to the Global Monte Carlo Method for Form-Factors Computation

Francesc Castro, Xavier Pueyo

Departament d'Informàtica i Matemàtica Aplicada
Universitat de Girona

Lluís Santaló s/n, 17003 Girona, Spain.

phone +34 (72) 418447/8763

fax +34 (72) 418399

e-mail {castro|xavier}@ima.udg.es

May 17, 1996

Abstract

For both local and global Monte Carlo methods for realistic rendering it does not exist hierarchical solutions. Some global Monte Carlo methods are based on the cast of random lines that pass through the whole scene. These lines are called *global lines*. Here we present a hierarchical improvement to this global cast based on the use of *locally global* lines, i.e. lines that pass through a subscene. In this way, we will take better information of the subscene. We will apply the improvement in a Monte Carlo global method to compute form-factors.

Keywords

Realistic Rendering, Radiosity, Global Monte Carlo.

1 Introduction

Radiosity [GTGB84] studies the exchange of luminic energy between surfaces within a closed environment. These surfaces must have diffuse reflection, and are divided into patches. The balance of energy flux in the environment is computed by solving a system of equations. In this system, form-factors play the main role. Form-factor is the fraction of energy flux leaving a patch that lands onto another patch. In the solution of the radiosity, the computation of the form-factors is often the bulk. Here we will deal with Monte Carlo methods to compute form-factors.

As well known, Monte Carlo methods, in opposition to analytical methods, do not compute exact values of form-factors, but estimators. These estimators are random variables. We consider local and global Monte Carlo methods. The first ones cast lines between every pair of patches. The global Monte Carlo methods involve the whole scene in the selection of rays. Here we will deal with the last ones, and specifically with methods that cast lines that pass through the scene.

These global methods have a drawback. Most of the scenes that we usually treat are rooms with some objects in their interior. So they are nearly empty rooms. Most of the global lines that these methods cast only intersect the walls, but not the objects. So these lines only contribute to the computation of form-factors *wall-wall*, not *object-object*, *object-wall* nor *wall-object*. We propose here an improvement that tries to avoid this drawback. The idea is to cast more lines where they are more necessary, and vice versa. So we will cast more lines in the zones with more density of objects. In this way, we can improve the computation of form-factors *object-object*, *object-wall* and *wall-object*. This idea involves a hierarchical approach. This is not the same as hierarchical radiosity, as presented in [HSA91], but it bears a hierarchy of subscenes where we will cast the lines.

This paper is divided in five sections. In Section 2 we will see, as a previous work, the Monte Carlo global method we will deal with. Section 3 presents the new method. Section 4 is the analysis of the results. Section 5 will briefly explain the conclusions to which we have arrived and some future work that has appeared from here.

2 Previous work

We will see here the global Monte Carlo method presented in [Sbe93] to which we will apply a hierarchical improvement. This global Monte Carlo method is based in *Integral Geometry*. Integral Geometry lets us do an analogy between measures of sets of lines and form-factors. So, as we can see in [Sbe93] and in [SP95], using Monte Carlo integration we can consider the form-factor F_{ij} between patch i and patch j as the probability of a line that exits patch i to land in patch j .

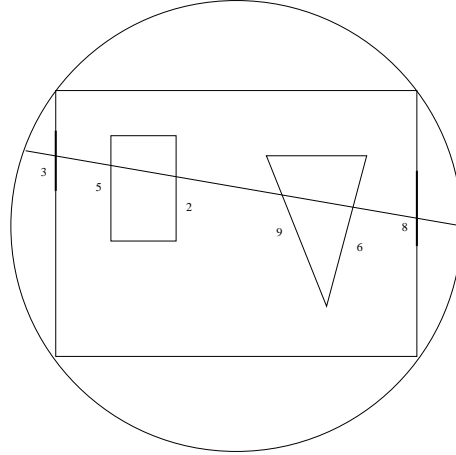


Figure 1: *Global method.* The visibility pairs are in this case 3 – 5, 2 – 9 and 6 – 8.

The method proposed in [Sbe93] uses Laplace’s Rule to compute this probability, namely, this probability is seen as the proportion of the lines that exit patch i and land in patch j . To estimate this proportion, this method casts global lines that pass through the whole scene. The scene is wrapped in a sphere, to which we will call *global sphere*, because it embodies the whole scene. Lines between pairs of random points on this sphere are casted.

For every line, we compute an ordered list of intersected patches. Then, we group the patches in the list of intersections in visibility pairs (figure 1). For every patch i in the scene, we have a counter of the number of intersections in it, called r_i . For every pair of patches (i, j) we also have a counter of number of lines that intersect both, called r_{ij} . We can easily see that $r_{ij} = r_{ji}$. Then, the estimator of form-factor F_{ij} is the ratio of the lines intersecting patch i that next intersect patch j :

$$\hat{F}_{ij} = \frac{r_{ij}}{r_i} \quad (1)$$

In this way, we obtain an estimator of the probability of a line that exits patch i to land in patch j , namely, an estimator of the form-factor F_{ij} .

3 Proposed method

3.1 Overview of the algorithm

The main idea is to avoid the drawback presented in the introduction: since most of the scenes are, in fact, nearly empty rooms, most of the global lines

casted only intersect with the walls, and so only contribute to the computation of the form-factors *wall-wall*. This problem appears in all the Monte Carlo global methods that cast global lines. We will try to avoid this problem with the application of a hierarchical improvement consisting of casting more lines in zones of the scene (subscenes) with more density of objects.

Now, we will not only use a global sphere that wraps all the scene but also local spheres that wrap only one or some objects of the scene. In the same way as we casted lines in the global sphere, here we will also cast them in the local ones. All the spheres will form a hierarchy. The algorithm to construct this hierarchy is based on a clustering method. Another important point is to distribute all the lines to cast among the spheres. We will apply heuristic criterions, that will be also presented.

The main points in this improvement are the update of the counters of number of intersections and, specially, the election of the form-factor estimator that we will use. The scheme of the algorithm is:

```

Create local spheres (a hierarchy)
Distribute lines among all the spheres (how many in each sphere)
Initialize counters of intersections
For every sphere
    For every line to cast in this sphere
        Cast random line
        Compute ordered list of intersected patches
        Update counters of intersections
    End For
End For
Compute form-factors

```

3.2 Form-factors estimation

We can estimate a form-factor F_{ij} (as seen in section 2) as $\hat{F}_{ij} = r_{ij}/r_i$, where r_{ij} is the number of lines that intersects patch i and patch j consecutively, and r_i is the number of lines that intersects patch i . To compute \hat{F}_{ij} we can use all the lines casted in spheres that contain patch i , because patch i is submerged in an uniform distribution with these lines; but we cannot use lines casted in spheres that do not contain patch i , because in this way the distribution would not be uniform for this patch. See figure 2. The distribution of the lines casted in the external sphere is uniform for both patches 1 and 2. But the distribution of lines casted in the internal sphere is only uniform for patch 1, that belongs to this sphere, but not for patch 2. For every line, we will group in pairs the ordered list of intersected patches and, for every pair (i, j) , we will proceed as follows:

- If both patches i and j belong to the sphere that casts the line, we will increase the counters r_i, r_{ij}, r_j and r_{ji} .

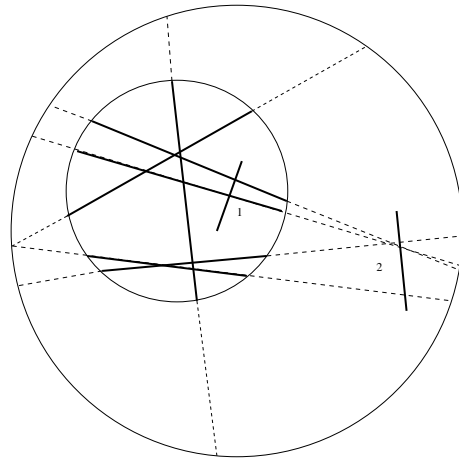


Figure 2: A *different-level casting of lines*

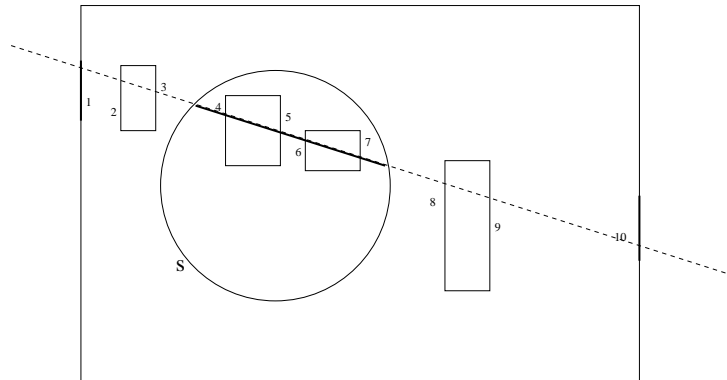


Figure 3: *Line casted in the inner sphere intersecting the whole scene*

- If only one patch, for instance i , belongs to the sphere that casts the line, we will only increase r_i and r_{ij} .
- If neither of the two patches belong to the sphere that casts the line, we will not increase any counter.

We can see an example of this in figure 3. If a line is casted in the sphere s , the pairs of patches 1 – 2 and 9 – 10 will not suppose any modification in the counters, because neither of these patches belongs to the sphere. The pair 5 – 6 will suppose a modification of all the counters, because both patches are inside the sphere. And in the pairs 3 – 4 and 7 – 8, we will only modify the counters relative to the patches 4 and 7, those that belong to the sphere.

Acting in this way, we can be sure of the next identity:

$$r_i = \sum_{\forall j} r_{ij}$$

But we cannot affirm that r_{ij} is the same as r_{ji} . The reason of this is clear: in our method the density of lines is not the same for every patch, and we only count intersections of patches that are inside the sphere that casts the line.

The other important aspect is to choose an appropriate estimator of the form-factors. Of course we can use the estimator proposed in [Sbe93]:

$$\hat{F}_{ij}^1 = \frac{r_{ij}}{r_i} \quad (2)$$

We can think about other estimators. In this new method, since the distribution of lines is not uniform inside the whole scene, because we cast partly in a local way, using the expression 2 the accuracy of \hat{F}_{ij}^1 and \hat{F}_{ji}^1 can be quite different. So, it can be interesting to use the well-known reciprocity relation:

$$F_{ij} = \frac{A_j}{A_i} F_{ji} \quad (3)$$

Then we can obtain a new estimator:

$$\hat{F}_{ij}^2 = \frac{A_j}{A_i} \frac{r_{ji}}{r_j} \quad (4)$$

So we can think in estimators obtained by some ponderations of \hat{F}_{ij}^1 and \hat{F}_{ij}^2 . We can use the number of lines that intersects every patch, r_i and r_j , as weights of every estimator. In this way we obtain:

$$\hat{F}_{ij}^3 = \frac{r_i \frac{r_{ij}}{r_i} + r_j \frac{A_j}{A_i} \frac{r_{ji}}{r_j}}{r_i + r_j} \quad (5)$$

We can see that the weight r_i only depends on patch i and the weight r_j only depends on patch j . They are not depending on the relation between both patches. If we use as weights the number of lines that leaves a patch and lands on the other one, namely r_{ij} and r_{ji} , then we have weights that are related at the same time with both patches, not with only one as was in the previous case. Now we have the following estimator:

$$\hat{F}_{ij}^4 = \frac{r_{ij} \frac{r_{ij}}{r_i} + r_{ji} \frac{A_j}{A_i} \frac{r_{ji}}{r_j}}{r_{ij} + r_{ji}} \quad (6)$$

N. OF LINES	ERROR \hat{F}_{ij}^1	ERROR \hat{F}_{ij}^3	ERROR \hat{F}_{ij}^4
$5 * 10^4$	1.1796	8.82378	1.1224
10^5	0.60453	4.27148	0.50391
$5 * 10^5$	0.11347	0.81667	0.09735
10^6	0.05634	0.44048	0.04850

Table 1: *Comparison of errors with different estimators of form-factors (scene NINECUBES)*

In table 1 we have a comparison of the proposed estimators. We have used a scene, called NINECUBES, that we will describe in the next section. The error that we present in the this table is a sum of squares of the differences between the exact value and the estimated value of the form-factors in the scene. This error is identified as FFE (form-factor error), and it will also be presented in the next section, the Analysis of results. Looking at the values in the table, we can see immediately that the second estimator, \hat{F}_{ij}^3 , offers results much worse that the others. The errors obtained with \hat{F}_{ij}^1 and \hat{F}_{ij}^4 look quite similar, and probably the differences can vary in tests with other scenes. So, it seems indifferent to use one or the other.

3.3 Spheres hierarchy

The kernel of the proposed method is the creation of the spheres hierarchy. To do this, we will apply in this first implementation a naif clustering procedure. We will make spheres at the object’s level: every object will have its own sphere, that will be the minimum enveloping one. Then, the algorithm will group in a new sphere the two spheres with minimum distance between their centers. This process will be repeated until we have a sphere which includes the whole scene. In this way, we create a binary tree of spheres, in which the leaf spheres will only contain one object. We can easily see that, if there are k objects, the number of spheres that will be created will be $2k - 1$. Here we must remark that the six walls will have a special treatment. They will not be considered as normal objects, namely, every wall will not have its own sphere. There will be a sphere, called global sphere, that will wrap the whole scene. This sphere will constitute the root of the tree (so, the total number of spheres will be $2k$, being k the number of "non-wall" objects. Figure 4 illustrates this method. A scheme of this clustering algorithm is:

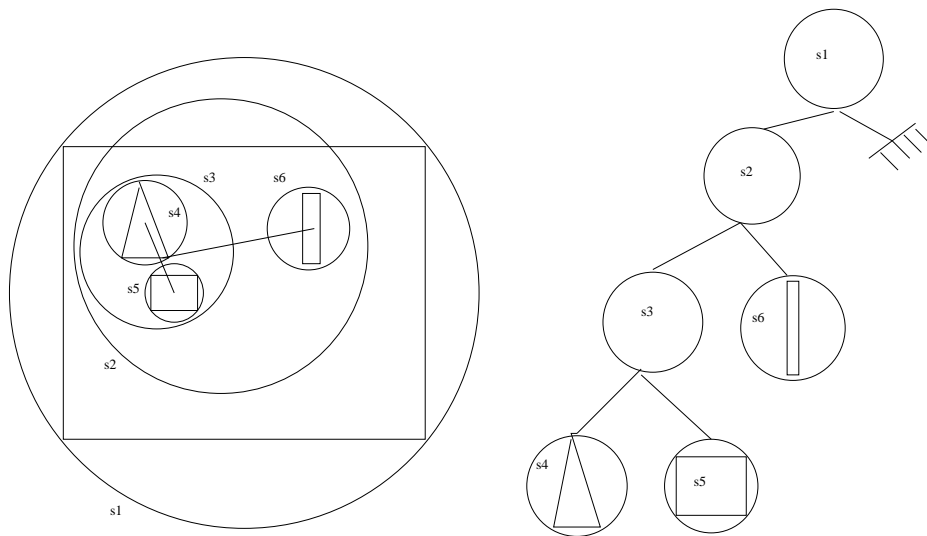


Figure 4: *Creation of the spheres hierarchy*

```

For every object in the scene (except the six walls)
    Make its minimum enveloping sphere
End For
For every pair of spheres
    Compute distances between their centers
End For
While exist two or more non-grouped spheres
    Group in a new (minimum) sphere the pair of spheres with minimum
    distance between their centers
    Remove the two grouped spheres from the table of distances
    Add the new sphere to the table of distances
End While
Add global sphere

```

For the distribution, among the spheres, of the lines to cast, we can apply some heuristics. Let us see a problem related with this question, that will lead us to a suitable heuristic. In figure 5 we can see a sphere that contains two objects. A lot of the lines casted in it will not intersect any object in the sphere. The more empty is the sphere, the more proportion of lines we will lose.

We will try to maximize the probability of a line to intersect any object in the sphere where the line is casted. First we will consider the simplest case: a sphere with only one object inside. In this case, Integral Geometry states that the probability of a line to intersect the object is given by the quotient between the area of the object and the area of the sphere (for convex objects). For more than one object, this probability is not the sum of the single probabilities, but

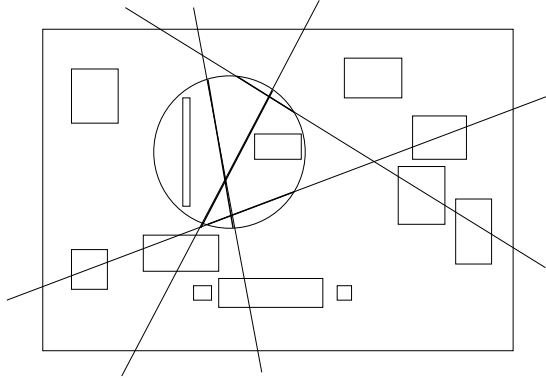


Figure 5: *Waste of lines casted in a local sphere*

we can use it like a coarse approximation to the exact value.

It seems reasonable to distribute the number of lines to cast according to these probabilities. So the number of casted lines will be, for every sphere, proportional to the quotient between the sum of the areas of the objects inside and the area of the sphere. An exception is the distribution for the main sphere. Since this sphere is special, because it contains the six walls, we will cast in it a constant proportion of the total number of lines (in our case, this proportion is a 20 per cent, and the remaining 80 per cent is distributed between the local spheres according to the explained criterion).

4 Analysis of results

The core of the method presented here is to cast lines in local spheres, with the goal of improving the Integral Geometry based method to compute form-factors from [Sbe93], a Monte Carlo global method that casts lines in only one global sphere that wraps all the scene. Then, it is very logical to approach the analysis of the results as a comparison with the referred method. This is what we will mainly present in this section. Since the method computes the form-factors, it seems reasonable to base this analysis on form-factor errors.

The selection of the scenes is crucial. Here we have chosen two simple scenes. The first one, called NINECUBES (figure 6), is composed by 9 cubes inside a cubic room. One of these cubes, the biggest, is placed near the ceiling. The other eight cubes have the same size, and they are placed on the floor, distributed in two groups of four cubes. The second one is called SIXCUBES (figure 7), and it is composed by 6 cubes. One of them, like in the scene NINECUBES, is near the ceiling, and the five remaining cubes are on the floor.

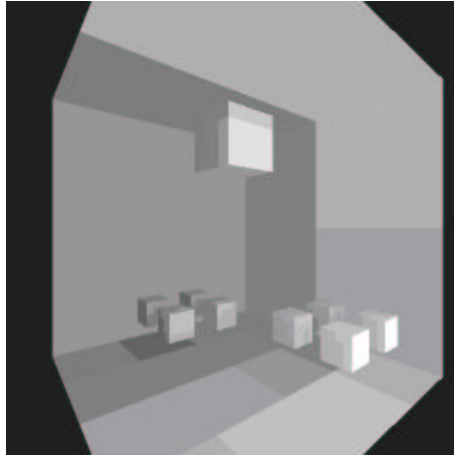


Figure 6: *Scene NINECUBES*

We would need an exact solution for form-factors, but it is not possible to find that. We will replace this exact solution by a “converged” one. To compute the “converged” solution, we have used a Monte Carlo local method for form-factors computation, casting a great number of lines (10.000 for every pair of patches). In this way, we have got very accurate form-factors.

4.1 Error in form-factors

At any rate we must consider that the accuracy of the form-factors does not exactly correspond to the accuracy of the radiosities of the scene. To compute radiosities - solving the system of equations - there are form-factors that are more important than others (for instance the form-factors from every patch to the source patches, etc). But to analyze the form-factors we will not consider this question, since form-factors are not depending on emissivities nor reflectivities of the patches.

We will define the form-factor Error (FFE) done by the next formula:

$$FFE = \sum_{\forall i,j} (\hat{F}_{ij} - F_{ij})^2 \quad (7)$$

where \hat{F}_{ij} is the estimate value for the form-factor and F_{ij} is the exact value (remember that we will use a converged value instead of this unknown exact value). We have studied three quantities for every execution done: the number of casted lines, the CPU time and the FFE. First we will present the results for the scene NINECUBES. These results are very significative. In table 2 we can see the results for the classical global method [Sbe93]. In table 3 there are the results for the improved new method. Finally we have the main result: in

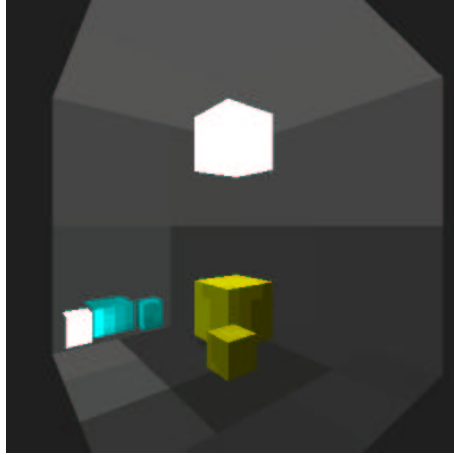


Figure 7: *Scene SIXCUBES*

N. OF LINES	TIME (seconds)	F.F. ERROR
$5 * 10^4$	49	4.0294
10^5	95	1.9018
$5 * 10^5$	463	0.37706
10^6	923	0.18986
$2 * 10^6$	1843	0.09987
$4 * 10^6$	3683	0.05151

Table 2: *Results for the scene NINECUBES with classical global method*

table 4 we show a comparison between the two methods. In figure 8 we plot number of lines and time. In figure 9 we have the representation of number of lines and FFE, and in figure 10 we plot time and FFE. With the same number of lines, the new method presents smaller errors than the classic one. Logically we must consider the CPU time, that, for the same number of lines, is greater in the new method. Anyway, looking at the curves we can see that the new method can offer a smaller error in the same time.

The behavior of the error in form-factors in the scene SIXCUBES is very similar to the behavior in the former scene NINECUBES. The corresponding tables (5, 6 and 7) and graphs (11, 12 and 13) show it clearly.

5 Conclusions and future work

We have presented a hierarchical improvement to the Monte Carlo global method presented in [Sbe93]. The carried out tests have shown that, for the tested scenes, we have got, with the same number of lines, better results

N. OF LINES	TIME (seconds)	F.F. ERROR
$5 * 10^4$	68	1.1224
10^5	132	0.50391
$5 * 10^5$	644	0.09735
10^6	1284	0.04850
$2 * 10^6$	2564	0.02690
$4 * 10^6$	5124	0.01651

Table 3: Results for the scene NINECUBES with hierarchy of spheres' method

N. OF LINES	CLASSICAL METHOD	NEW METHOD
$5 * 10^4$	4.0294	1.1224
10^5	1.9018	0.50391
$5 * 10^5$	0.37706	0.09735
10^6	0.18986	0.04850
$2 * 10^6$	0.09987	0.02690
$4 * 10^6$	0.05151	0.01651

Table 4: Comparative of the errors with both methods for the scene NINECUBES

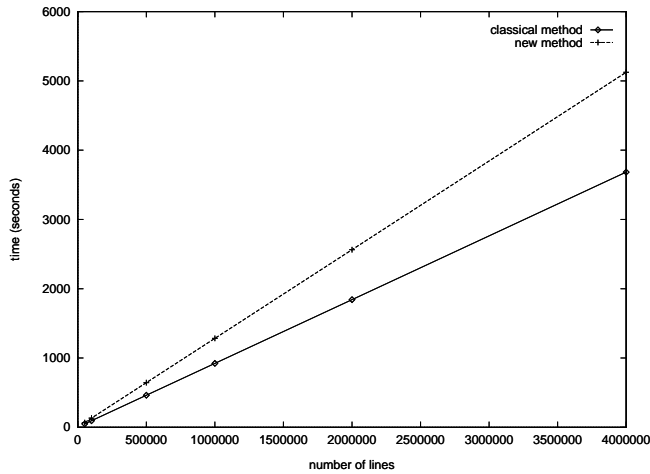


Figure 8: Scene NINECUBES. Time spent by several executions (with different number of lines) in both methods

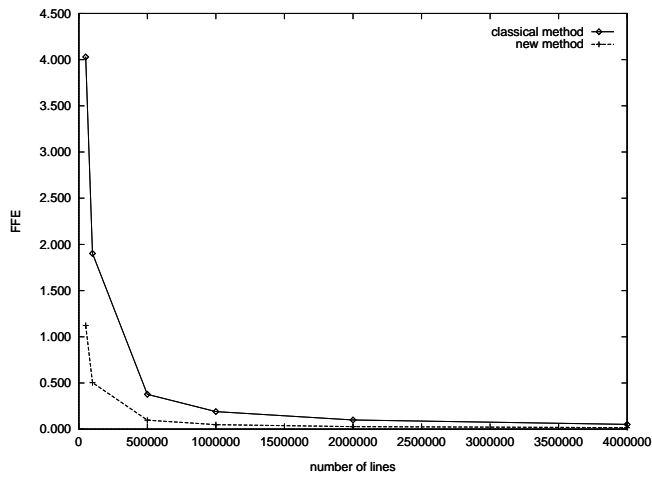


Figure 9: Scene *NINECUBES*. FFE in both methods for different number of lines

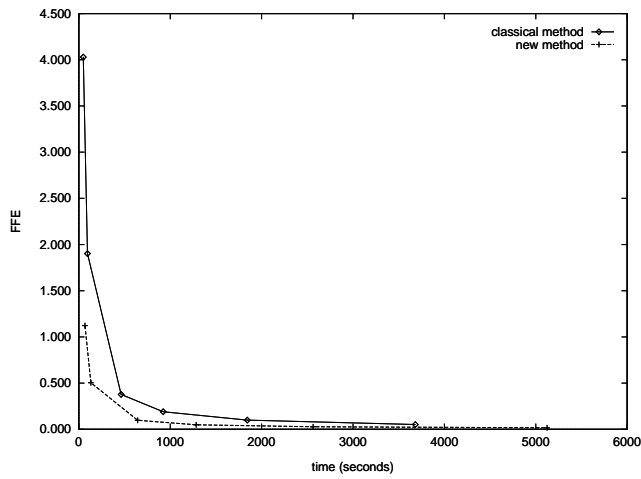


Figure 10: Scene *NINECUBES*. FFE in both methods versus time

N. OF LINES	TIME (seconds)	F.F. ERROR
$5 * 10^4$	47	2.1563
10^5	92	1.0776
$5 * 10^5$	452	0.23079
10^6	902	0.11681
$2 * 10^6$	1802	0.06892
$4 * 10^6$	3602	0.04235

Table 5: Results for the scene *SIXCUBES* with classical global method

N. OF LINES	TIME (seconds)	F.F. ERROR
$5 * 10^4$	66	0.53527
10^5	130	0.24104
$5 * 10^5$	642	0.05680
10^6	1282	0.03704
$2 * 10^6$	2562	0.02703
$4 * 10^6$	5122	0.02200

Table 6: Results for the scene *SIXCUBES* with hierarchy of spheres' method

N. OF LINES	CLASSICAL METHOD	NEW METHOD
$5 * 10^4$	2.1563	0.53527
10^5	1.0776	0.24104
$5 * 10^5$	0.23079	0.05680
10^6	0.11681	0.03704
$2 * 10^6$	0.06892	0.02703
$4 * 10^6$	0.04235	0.02200

Table 7: Comparative of the errors with both methods for the scene *SIXCUBES*

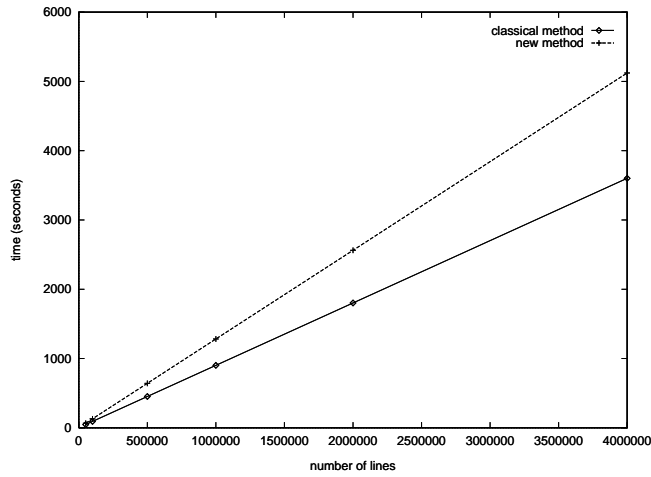


Figure 11: Scene *SIXCUBES*. Time spent by several executions (with different number of lines) in both methods

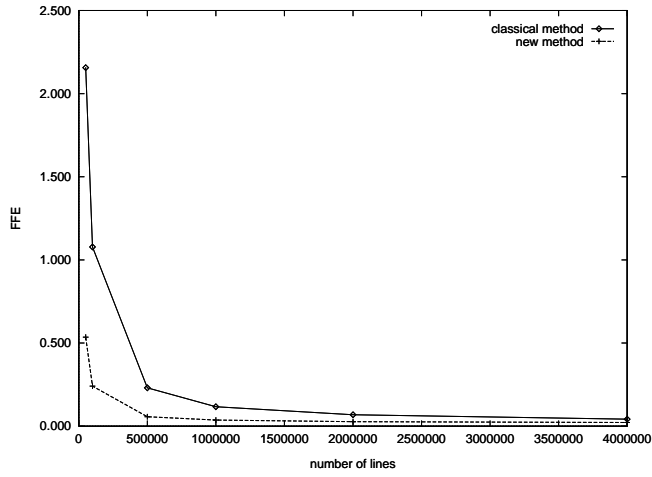


Figure 12: Scene *SIXCUBES*. FFE in both methods for different number of lines

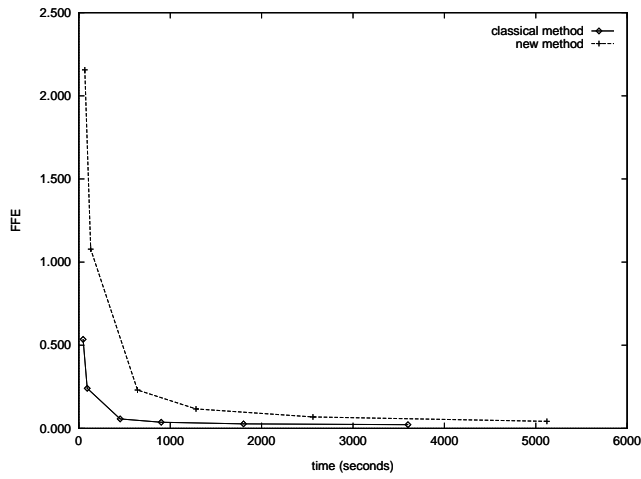


Figure 13: Scene *SIXCUBES*. FFE in both methods versus time

for form-factors with the new improvement. We have also studied the CPU time instead of the number of lines, and we have obtained similar conclusions, although the CPU time is very dependent of the used coherence. Furthermore, it exists a strong dependence between the results and the choosen scenes. So, a first future work should be to extend the number of scenes (using, for instance, simple scenes to study the influence of concrete factors as disposition, size or form of the objects).

It would be interesting to do a more formal analysis of the different form-factor estimators that we can use, and even to propose new estimators. In this report, the evaluation of the estimators has been completely empirical. In the other hand, the coherence applied could be improved, by searching a coherence system more suitable to this method. Another question would be the appliance of a first shoot of primary sources, that improves in a important way the Monte Carlo global methods. Another pendent question would be a more formal treatment of the used form-factors error measure. Here we have treated it in a intuitive way. It would be possible to propose other error measures.

A possible variation of the proposed method would consist of treating only the intersections of the lines with the objects inside the sphere where these lines have been casted. In this way, although we would lose information, we will save intersection time. This question is very related with the use of an appropriate type of coherence. Finally, we could apply a similar hierarchical idea to the Multi-Path method presented in [SPNP95].

6 Acknowledgements

Many thanks to Gonzalo Besuievsky, Narcís Coll, Josep Daunis, Frederic Pérez and Mateu Sbert for their useful comments in many instants during all this work, and to Jordi Regincós for helping with LATEX with which we have written this paper. This project has been funded in part with grant number TIC 95-0614-C03-03 of the CICYT.

References

- [GTGB84] C.M. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *Computer Graphics (ACM SIGGRAPH Conf.Proc.)*, 18, N.3:213–222, 1984.
- [HSA91] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *ACM Computer Graphics*, 25(4):197–206, 1991.
- [Sbe93] M. Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum (proc. Eurographics'93)*, 12, N.3:409–420, 1993.
- [SP95] M. Sbert and X. Pueyo. Integral geometry methods for form-factor computation. *Proc. of the VI Encuentros de Geometría Computacional*, pages 297–305, 1995.
- [SPNP95] M. Sbert, X. Pueyo, L. Neumann, and W Purgathofer. Global multi-path monte carlo algorithms for radiosity. *The Visual Computer*, 1995.