Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors

Bruno Arnaldi, Thierry Priol, Luc Renambot, Xavier Pueyo

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors

Bruno Arnaldi, Thierry Priol, Luc Renambot, Xavier Pueyo

N ° 3008

Octobre 1996

THÈMES 1 et 3

ISSN 0249-6399

INRIA
RENNES

Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors

Bruno Arnaldi*, Thierry Priol*, Luc Renambot*, Xavier Pueyo †

Thèmes 1 et 3 — Réseaux et systèmes — Interaction homme-machine, images, données, connaissances
Projet Caps et Siames

Rapport de recherche n°3008 — Octobre 1996 — 22 pages

Abstract: In this paper, we present a modified version of the virtual wall concept we introduced in a previously published paper. The goal of our work is to design a strategy to handle very complex scenes (more than 1 million of patches) for radiosity computation. Comparing to other radiosity algorithms, our solution focuses on the ability to compute the radiosity on local environments instead of solving the problem for the whole environment. By splitting the problem into subproblems, using Virtual Interface and Visibility Masks, our technique is able to achieve better data locality than other standard solutions. This property is capital when using either a modern sequential computer to reduce data movement in the memory hierarchy or a multiprocessors to keep as low as possible communication between processors whatever the communication paradigm is: either message passing or shared variable. In the paper, we present an implementation of visibility masks on a distributed memory parallel computer (Intel Paragon XP/S).

Key-words: radiosity, complex databases, parallelism, data locality.

(Résumé : tsvp)

* {arnaldi,priol,renambot}@irisa.fr
† Girona University, Girona, Spain, xavier@ima.udg.es

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Masques de visibilité pour effectuer des calculs complexes de radiosité sur machine multiprocesseurs

Résumé : Dans ce article, nous présentons une version modifiée du concept des murs virtuels que nous avons introduit dans une publication précédente. Le but de notre travail est de concevoir une stratégie pour traiter des scènes très complexes (plus d'un million de polygones) par une méthode de radiosité. Par rapport à d'autres algorithmes de radiosité, notre solution porte sur la possibilité de calculer la radiosité sur des sous-environnements et non sur la scène dans son ensemble. Par la division du problème en sous-problèmes, en utilisant les notions d'Interface Virtuelle et de Masque de Visibilité, notre solution, comparée à d'autres techniques classiques, apporte une meilleure localité des données. Cette propriété est essentielle avec l'emploi des ordinateurs séquentiels modernes pour réduire les mouvements de données dans la hiérarchie mémoire, ou dans un contexte multiprocesseurs pour maintenir les communications entre processeurs à un niveau le plus faible possible, quelque soit le modèle de communication : par échange de messages ou par variables partagées. Nous présentons dans cet article une implémentation des masques de visibilité sur une machine parallèle à mémoire distribuée (Intel Paragon XP/S).

Mots-clé : radiosité, scènes complexes, parallélisme, localité des données.

1 Introduction

Over the past ten years, radiosity has become more and more popular since this technique is able to produce realistic images taking into account ambient light reflected between objects in the scene. Several works have been carried out to increase the correctness and the effectiveness of the algorithms to solve the radiosity problem. Concerning the speeding-up of radiosity algorithms, both sequential and parallel solutions have been proposed. This includes progressive radiosity, hierarchical solutions, importance driven and clustering. Unfortunately, despite numerous improvements, radiosity algorithms generate a huge number of computations and have a lack of data locality. This latter drawback makes the execution of radiosity algorithms ineffective on either uniprocessor or multiprocessors computers with a complex memory hierarchy as soon as the problem size (the number of patches) is huge. This problem was not so sensitive with other rendering algorithms such as scan-line, Z-buffer or ray-tracing. For them, a data locality property exists and makes the execution of such algorithms relatively effective by using the memory hierarchy properly.

To illustrate this, let's take the ray-tracing algorithm. The computation of two neighboring pixels will have a great probability to generate roughly the same data access pattern since rays will probably be cast in the same direction. On a uniprocessor, the computation of the second pixel will thus use most of the data located in cache due to the computation of the first pixel. For multiprocessors, the same property can be used to keep low the number of data accesses to the shared memory (either real or virtual). This has been clearly demonstrated by the work of Green et al.[11] or Badouel et al.[2].

Since the introduction of the radiosity, a lot of works have been carried out to decrease the computation time either by optimizing the algorithm by reducing the number of computations or by using high performance computers such as those that have several dozen of processors. However, few of these solutions dealt with the improvement of data locality. In [28, 25, 1], several ideas have been proposed to deal with complex environment. They are mostly based on Divide and Conquer strategies: the complex environment is subdivided into several local environments where the radiosity computation is applied.

In this paper, we propose an new improvement to the virtual wall concept we have already introduced some years ago [1]. This improvement deals with the energy transfer between two local environments. We introduce a new mechanism we called *visibility mask* which aims at representing the flow of energy that is exchanged between local environments.

The paper is organized as follows. First, section 2 describes some other works which seem similar to our technique. Section 3 presents the virtual interface concept with the *visibility masks*. Section 4 gives a parallel algorithm we designed mainly for distributed memory parallel computers. Section 5 provides some performance measurements. Conclusion and perspectives are presented in section 6.

2 Related works

As we stated, the target of our work is to propose an efficient radiosity algorithm able to deal with very complex scenes. This goal is achieved by means of a technique which exploits data locality. This technique can be projected onto both sequential and parallel machines. So we will analyse previous work from two points of view: parallel solutions and, mainly, locality enhancement.

A number of parallel solutions have been proposed using different architectures and different types of parallelising strategies. In [13], [4], [10] discussions are presented showing the possible approaches for parallelising radiosity at several levels. From the scope of the work presented here, we may distinguish between two kinds of solution. A first family of algorithms allows the access of all the processors to the entire database. This may be propitiated by shared memory systems or by distributed memory systems where the database is duplicated at each local memory [3], [16], [15]. In the rear case, scene's size is very limited. The second family of algorithms is constituted by the techniques based on distributed memory systems where the database is distributed among the different local memories allowing the application of the algorithm to bigger scenes [7, 12, 26, 6, 27, 23]. Solutions based on distributing the environment among the local memories are often based on an explicit or implicit subdivision of the environment in order to capitalize on any type of data locality to reduce the exchange of information between processors. To the authors knowledge, none of these methods exploits data locality based on both: minimising the amount of information transmited between processors and taking advantage of the visibility information implicit in the environment's decomposition.

The idea of considering sub-environments (or local environments) has been used from different points of view, in order to reduce computational cost in radiosity algorithms for complex environments. The first contribution in this direction was proposed in [28]. This method divides environment into local environments and computes form-factors in each of them. Afterwards, form-factors between patches in neighbouring local environments are computed from previously computed local form-factors. Similar approaches were presented in [25] and [1]. Energy is accumulated at the boundaries of neighbouring local environments (called virtual walls) and afterwards transfered. Instead of dividing the environment into smaller ones, [14] proposes to group neighbouring patches. These groups receive and shoot energy as one single entity. We may call them virtual surfaces.

Groups of surfaces are used in order to simplify energy exchange in complex environments by means of simplifying the geometry of the environment in a 3-D grouping approach in [19], [20], [22] and [21]. We understand by 3-D grouping, the fact that these methods define 3-D clusters (or boxes) instead of groups of patches with similar orientation as in done in [14]. Then energy exchange takes place between clusters. The algorithms may or may not include a hierarchical strategy. Clustering is also used in [17] where the environment is mapped onto the walls of the cluster. This map is used only for secondary rays while primary rays do not use the clusters. The interest of parallelising the algorithm is pointed out in [17] because of the locality resulting from the mapping but this locality is only exploited during

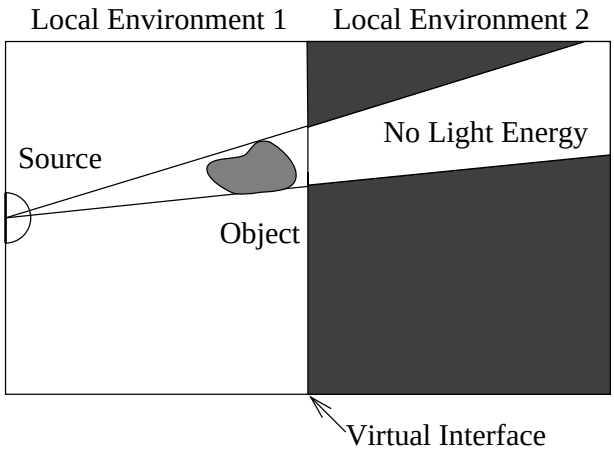


Figure 1: Virtual Interface (former Virtual Wall).

the rendering step. All these methods introduce approximations for exploiting locality but bound their effects in order to keep the required accuracy.

In [24] is proposed a system to exploit locality in order to minimize the transfers between disk and main memory. This system is designed to compute radiosity efficiently in very large environments where the visibility relationships are locally dense but globally sparse. It is based on a hierarchical approach. Two basic ideas are exploited: partitioning and ordering. Partitioning means identifying subsets of the database each composed by the clusters which interact with a given (receiver) cluster. The definition of these subsets is based on the scene structure and a visibility precomputation between clusters. Ordering implies taking the subsets in an order which minimizes the difference between two successive subsets. The results in [24] show that, for the specific type of environment described above, the best efficiency was obtained with cell order. These cells are defined by a wall-aligned BSP-tree spatial subdivision.

3 Virtual Interface

Virtual interface is an improvement of the virtual wall concept we first introduced in [1]. The basic idea of this work was to find a way to subdivide a complex environment in local environments using planes we called *virtual walls*.

The virtual wall is a mechanism which aims to gather and to scatter the energy leaving or entering a local environment. However, the use of virtual walls has some disadvantage since it requires additional computations and memory. As for the object of the scene, a virtual wall has to be decomposed into patches for which the energy going through has to be stored. Representing the energy leaving a patch from a virtual wall requires the storing of

several information such as the spatial distribution of the energy. Moreover, the subdivision of a virtual wall has to be done in such a way that it has few impact on the quality of the rendering. To overcome these problems, we introduce a new technique called Virtual Interface which is described in the following paragraphs.

With this new technique, a virtual interface is no more a set of patches. Instead of having these patches to represent the exchange of energy between local environments, we use a new mechanism, called Visibility mask to transfer light energy between local environments. The aim of this mechanism is to allow a local computation of the radiosity in a local environment in order to improve the data locality access.

3.1 Overview of the virtual interface

The first principle of the virtual interface addresses the energy transfer between local environments. Instead of accumulating the energy from different sources onto a *virtual wall* before sending it to another processor, the *virtual interface* technique allows the management of a source separately from other sources in the same local environment. As soon as a source has emitted its energy, it is sent to the next processor.

The second principle of the virtual interface concerns the way to take into account the source transfer between local environments. Indeed, sending the geometry and its emissivity to the next environment after a local processing imply to add a new structure, called **visibility mask**, to the source (figure 1). The visibility mask stores, in the source structure, all the occlusions encountered during the processing in each local environment.

The next sections describes more deeply the two principles.

3.2 Visibility mask

To describe more deeply our technique, we will take the assumption that a progressive radiosity technique is used jointly with a hemisphere to compute form factors. With such technique, a patch which has the most unshot energy is selected and its energy is distributed to other patches in the scene according to a form factor computation. With our virtual interface concept, the energy of each selected patch, we call it a source, is first distributed in the local environment in which it is contained. Then, its energy is propagated to other local environments. However, to propagate efficiently the energy of a given patch to another local environment, it is necessary to determine the visibility of the patch according to the current local environment: it may exist an object along a given direction which hides the source (figure 2). We introduced the visibility mask which is a subsampled hemisphere identical to the one involved in the computation of the form factors. Each pixel of the hemisphere, that is used for the form factor computations, corresponds to a boolean value in the visibility mask. A value is set to false if the corresponding ray, going through the pixel of the hemisphere, hits an object in a previous local environment. Note that our technique can be applied to hemicube, in that case the visibility mask will be an hemicube as well. Moreover, it can be easily extended to take into account participating media. Instead of using boolean values, an integer value could represent the percentage of absorption along a given direction.

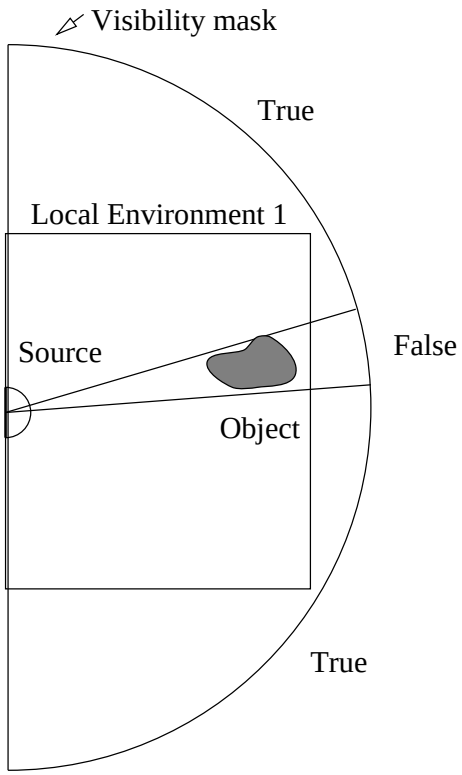


Figure 2: Initialisation of the visibility mask for a local source.

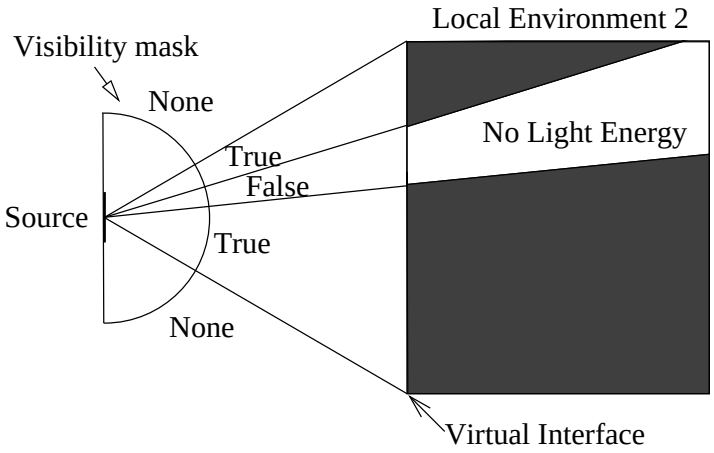


Figure 3: Use of the visibility mask for a distant source.

3.3 Processing of visibility masks

The visibility mask allows the distribution of energy to local environment in a step by step basis. Two cases may arise. If the source belongs to the local environment, a visibility mask is created and all its boolean value is set to `true` otherwise the visibility mask already exists and will be updated during the processing of the source. Form factors are computed with the patches belonging to the local environment by casting rays from the center of the source through the hemisphere. If a ray hits an object in the local environment, the corresponding value in the visibility mask is set to `false` otherwise there is no modification. Then, radiosities of local patches are updated using one iteration of a progressive radiosity algorithm. Finally, the source and its visibility mask are sent to the neighbouring local environments to be processed later (figure 3).

3.4 Space subdivision with virtual interfaces

One of the main difficulty in using virtual interfaces is to find a mapping of virtual interfaces in the complex environment. Due to the computation of the visibility masks, the processing of a source cannot be carried out in any order. Indeed, a visibility mask contains visibility information from all other local environments between the source and the current local environment. Figure 4 shows both a 1D and a 2D space subdivision. Regarding the 1D case, before processing a source in LE_3, it has to be processed in LE_1 and LE_2 in a sequential manner. If the target computer has several processors, there is no way to exploit them simultaneously in the processing of one source which such space subdivision. With a 2D decomposition, the processing of a source has to be done using a wavefront technique allowing the computation of the source in parallel. Local environments belonging to the

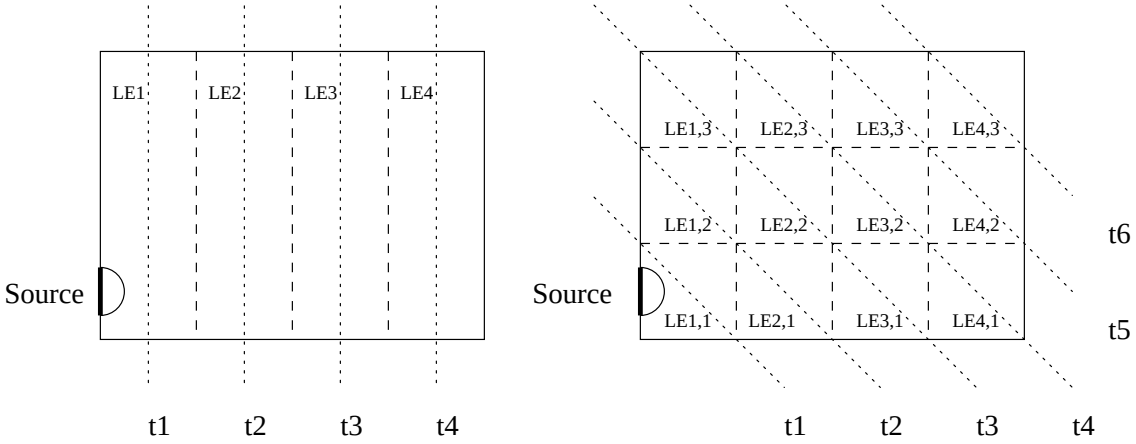


Figure 4: 1D and 2D space subdivision with virtual interfaces

same wavefront t_x can be thus evaluated in parallel. This property can be extended for a 3D subdivision of the initial environment. Such data dependencies, in the processing of sources, could be represented naturally by mean of messages in a parallel implementation of the *virtual interface* concept.

4 Distributed memory implementation

The design of a parallel algorithm for distributed memory parallel computers requires first and foremost a data domain decomposition in order to distribute data among the local memories. This first step is then followed by a distribution of computations according to data distribution. Computations are assigned to a processor that owns the data needed to performed them. Such technique has been widely used, it is known as data oriented parallelisation. The virtual interface concept, we developed in this paper, allows us to apply this data oriented parallelisation scheme to the radiosity computation. The initial environment is splited into several local environments which can be mapped on the available processors. A subset of these local environments is assigned to a processor. A processor is in charge of performing the computation related to its own data. It consists in the distribution of the energy of patches that belong to its local environments as well as the distribution of energy coming from neighbouring local environments.

Before going into details, we describe briefly, in the next section the hardware and software environment we are using for experimenting our parallel radiosity algorithm. The description of our solution is given in section 4.2.

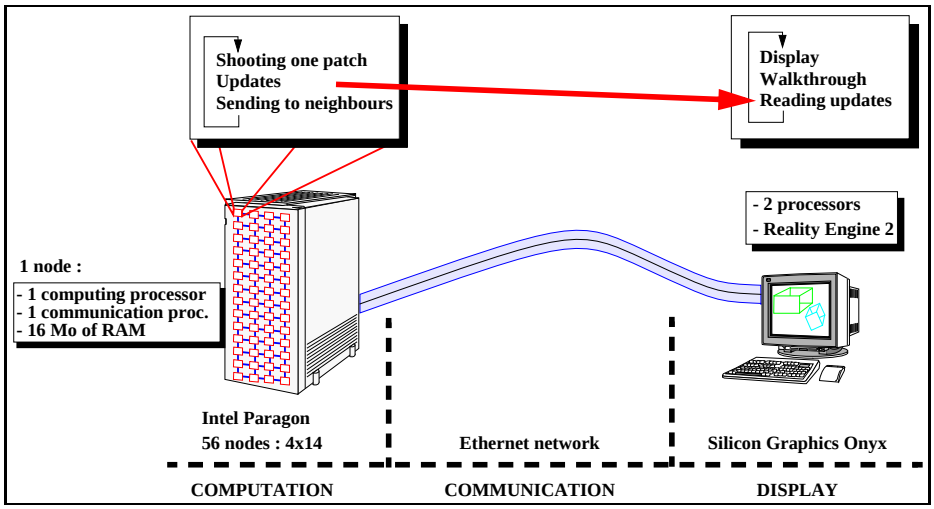


Figure 5: Overview of the experimental platform.

4.1 Experimental platform

We began our work by designing an experimental platform. This platform is made of several parallel computers which allow computation and visualisation to be done simultaneously. Figure 5 shows an overview of the experimental platform. Radiosity computations are performed in parallel by an Intel Paragon XP/S having 56 compute nodes and 2 service nodes. Radiosity updates are gathered from computes nodes by a service node located on the Paragon. Then, the service node sends them, through an Ethernet network, to a 2 processors Silicon Graphics Onyx for visualisation purpose. Figure 6 gives a short description of the three algorithms running on the two parallel computers.

On the Paragon side, a process (*ServiceNode*) is created on one of the service node. This process forks immediately one child process (*Virtwall*) on each compute node, then it waits until it receives radiosity updates from the compute nodes. These updates are sent to the *Display* process running on the Onyx. The *Display* process exploits the **Performer** library provided by SGI [18]. Both the Paragon and the Onyx have a copy of the database therefore the amount of communication, exchanged during the computation between the two machines, is kept small. On the Onyx side, users may interactively walk through the image or may focus on one local environment associated to a particular processor while the image is being displayed.

Figure 7 and 8 present two images displayed on the Onyx system connected to the Paragon XP/S. The first image shows the **Performer** data tree where each leaf represents a local environment. Users may select this leaf to display the portion of the image associated with the local environment (Figure 8).

Paragon XP/S		Onyx

ServiceNode()
{
fork_on_all_nodes(**Virtwall**);
socket_open(display);
 	
while (! end)
{
receive_from_node(update);
socket_send(update);
}
 	
socket_close();
}
	Virtwall()
	{
	load_scene();
	
	while (! converged)
	{
	choose_best_source();
	compute_illumination();
	send_source_neighbours();
	send_to_host(update);
	}
	
	}
		Display()
		{
		load_scene();
		socket_create();
		while (! simdone)
		{
		process_user_event();
		sock_recv_non_blocking(update);
		compute_frame(update);
		display_frame();
		}
		socket_close();
		}

Figure 6: Algorithms running on the Paragon XP/S and the Onyx parallel computers.

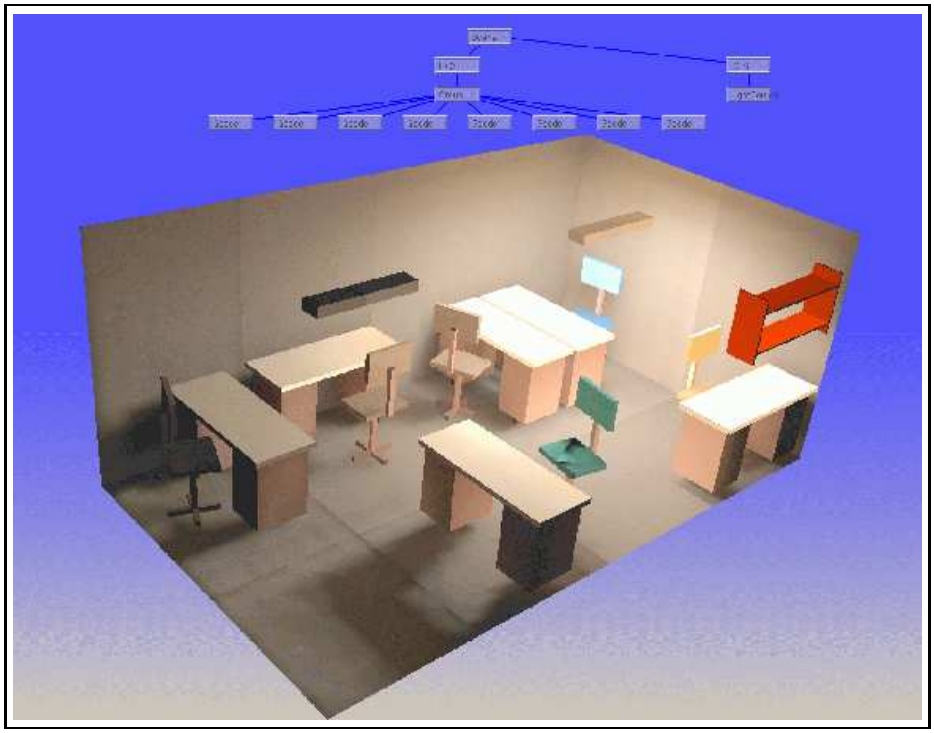


Figure 7: Final image with the *Performer* tree at the top.

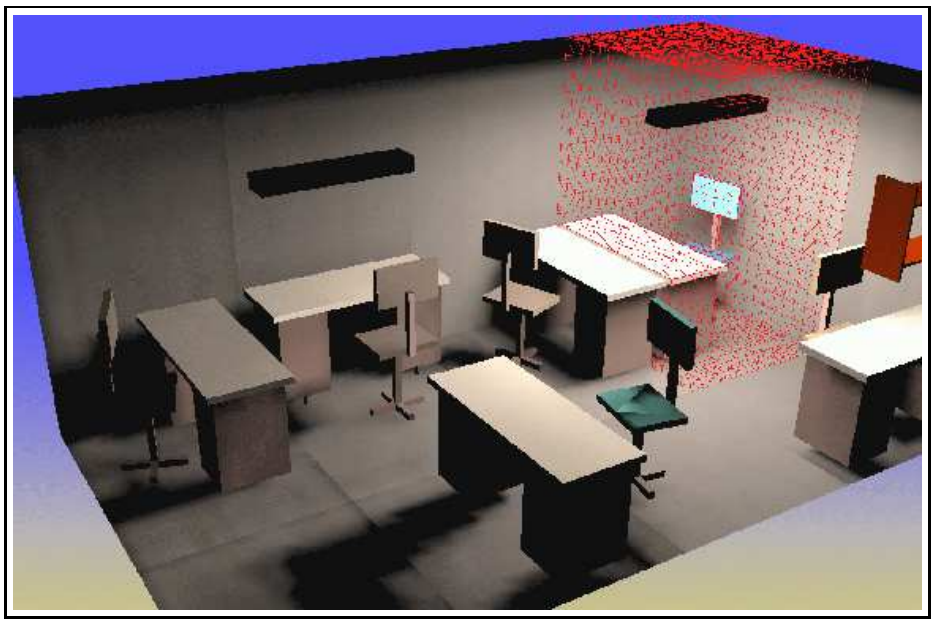


Figure 8: Image partially computed with a local environment associated with a processor.

```
void ComputeNode()
{
    Initialisation()
    do {
        do { /* Computing */
                Choose source among region and network queue;
                Shooting the source;
                Read all the sources arrived from network, put them in queue;
        } while(! local_convergence);
        TerminationDetection(done)
    } while ( not done);
}
```

Figure 9: Algorithm running on the compute node.

4.2 Parallel algorithm description

The parallel algorithm running on each compute node of the Paragon consists in three successive steps: an initialization, a radiosity computation and a termination detection as shown in figure 9. These three steps are described in the following paragraph.

4.2.1 Initialisation

Each compute node performs an initialisation step which consists in reading the local environment geometries that have been assigned to it. The specification of the local environment geometries as well as their assignment to a particular compute node is performed automatically but straightforwardly in the current version of the algorithm. Since such operation is relatively complex and has an impact on load balancing, we plan to perform this operation using an appropriate strategy in a near future. Once each compute node has the description of its local environment geometries, it reads the scene database to extract polygons that belongs to its local environments. Such strategy represents a bottleneck since all the compute nodes perform I/O operations at the same time. However, it has the advantage not to statically split the scene database into several files for a given number of processors. This is a substantial advantage when several experiments have to be performed with different environment decomposition on various number of processors.

A polygon that belongs to several local environments is splited into as many part as the number of local environments it belongs to. Such policy simplifies the processing of patches during the radiosity computations. Afterwards, polygons are decomposed recursively into patches. Each polygon is a quad-tree of patches.

The last processing step consists in subdividing the local environment into cells using a regular spatial subdivision [8] in order to speedup the ray-tracing process when form factors are computed.

4.2.2 Patch selection

Once the initialization has been performed, each compute node selects an emitter patch which has the greater delta-radiosity from either its local environment or from its receive queue which contains patches and visibility masks that have been sent by other compute nodes since the last patch selection. At the beginning of the computation, these queues are empty; therefore, each compute node selects a local patch even if its energy is small compare to patches located in the other local environments. However, such behaviour increases the number of iterations before reaching the convergence.

4.2.3 Radiosity computations

As explained in section 3, each selected patch is associated with a visibility mask that represents the distribution of energy. Initially, when a processor selects a patch which belongs to its local environment, the visibility mask is set to true. As the form-factor calculation progress, using a ray-tracing technique, part of the visibility masks is set to false if the rays hit an object in the local environment. Once the energy has been shot in the local environment, it is necessary to determine if there is still some energy to be sent to the neighbouring local environments. A copy of the visibility mask is performed for each neighbouring local environment. A ray-tracing is then performed for all pixels which have a true value. Intersection with the considered plane is performed to determine if there is energy entering in the neighbouring local environment. If a ray hits the plane that separates the two local environments, the corresponding pixel of the visibility mask is left unchanged otherwise it is set to false. At the end of this process, if the copies of the visibility mask have still some pixel values to true, they are sent to processors which own the neighbouring local environments together with some information related to the geometry and the photometry of the emitter patch. The message size is approximatively $4KBytes$. The most important part of the message is used to store the visibility mask that is encoded using one bit for each pixel.

4.2.4 Termination detection

Each compute node performs its computation independently from the other compute nodes. Therefore, it has no knowledge about the termination of the other compute nodes. Termination detection is carried out using two steps.

The first step consist in deciding to stop the selection of a local patch if its energy is under a given threshold. The service node is in charge of collecting, on a regular basis, the sum of the delta radiosities of the local environments as well as the energy that is under transfer between compute nodes. Depending of this information, the service node can inform compute nodes to stop the selection of new local patches.

Once the first step is reached, buffers that contain sources coming from other compute nodes have to be processed. To detect that all the buffers are empty, a termination detection is carried out using a distributed algorithm based on the circulation of a token [5]. Compute

nodes are organized as a ring. A token, whose value is initially true, is sent through the ring. If a compute node has sent a patch to another node in the ring since the last visit, the value of the token is changed to false. It is then communicated to the next compute node. If the compute node, that initiates the sending of the token, received it later with a true value, it broadcasts a message to inform all the compute nodes that the radiosity computations have ended.

5 Experiments

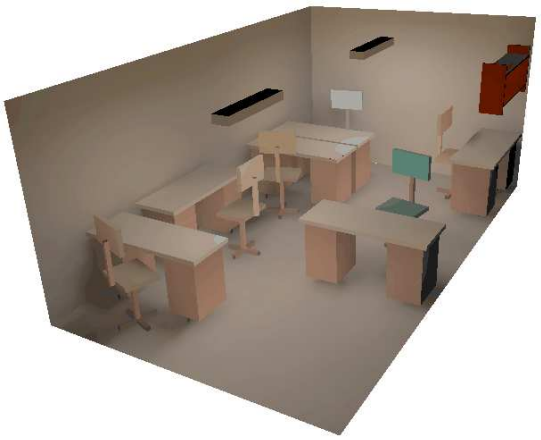
Several experiments have been performed to study the behavior of our radiosity algorithm running either on a uniprocessor or a multiprocessors. Experiments using a uniprocessor workstation was devoted to the analysis of potential overhead due to the use of virtual interfaces and visibility masks comparing to a traditional approach. Other experiments were performed on a Paragon XP/S using the parallel algorithm as described in the previous sections. Prior to the presentation of these results, the following section gives the description of three databases that were used in the experiments.

5.1 Scene description

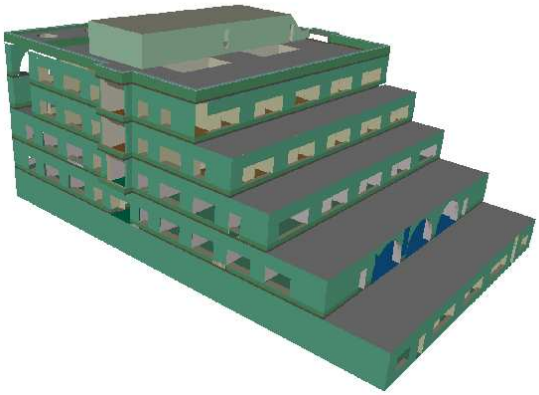
Experiments were performed using three different scenes (10). The *Office* scene represents an office with tables, chairs and shelves. The scene contains two lights on the ceiling. It's an open scene with few occlusions. It is made of roughly six hundred polygons. After meshing, 7440 patches were obtained. The second scene is a set of 32 similar rooms. Four tables, four doors open onto next rooms and one light source compose a room. This is a symmetrical scene with many occlusions. This file comes from benchmarks scenes presented at the 5^{th} *Eurographics workshop on rendering*. After meshing 17280 patches were obtained. The last scene is the biggest one. It represents five floors of a building without any furniture and with one thousand light sources. This scene represents the *Soda Hall Berkeley Computer Science* building [9]. After meshing, 71545 patches were generated.

5.2 Experiments on a uniprocessor

The first experiment was performed in sequential using a Sun UltraSparc workstation. The goal is to show that the use of virtual interfaces and visibility masks do not imply a huge software overhead comparing to a more traditional approach. The three scenes were rendered using several local environment decompositions (from 1 to 56 local environments). One local environment corresponds to the traditional approach since there is no virtual interface and visibility mask to add overhead. In these experiments, the number of cells is identical whatever the number of local environments is. Consequently, cutting planes of local environments must lay on the spatial subdivision grid. The sequential algorithm processes local environments ones after the others in a sequential order. During one iteration, all sources, waiting in the queue of the current local environment, are shot.



a) Office



c) Building

Figure 10: Scenes pictures

Results, using the three scenes with different number of local environments, are shown in figure 11. Time in seconds is shown in the left axis whereas the right axis shows the corresponding gain in relation to the computation time of the traditional approach (one local environment).

As shown in figure 11, the performance of our technique depends on the scene. For a small scene like *Office*, there was no gain; computations time was greater than the traditional approach. With 16 local environments, computing time increases due to a greater number of shoots. However, as the number of polygons increases, we got much better performance. This increasing performance can be explained by the decreasing complexity of some parts of the radiosity algorithm which depends on the number of polygons. When using a spatial subdivision to generate local environments, the number of polygons is reduced in each local environment and thus complexity is reduced. Selection of the most emissive patch and radiosities update can be performed more rapidly. Moreover, a smaller working set implies a better behavior of the different level of the memory hierarchy: primary cache, secondary cache, virtual memory). We are currently performing experiments to evaluate the impact of the memory hierarchy.

5.3 Experiments on a multiprocessors

The second kind of experiments was performed using the Paragon XP/S. Since the computing time of the sequential version depends on the number of local environments, we took a decomposition of the scene into 56 local environments in order to avoid super-linear speedup. As said previously, decomposition is a straightforward automatic process without optimisation to balance the load among the processors. Computing times and speedups are are shown in table 1 and synthesized in figure 12. Despite that no effort was spent to solve the load balancing problem, speedups were quite good comparing to other parallelization strategies previously published.

	Office		Rooms		Building	
Procs.	Time	Speed-up	Time	Speed-up	Time	Speed-up
1	147.58	1.00	1924.44	1.00	6501.08	1.00
2	85.42	1.73	1231.30	1.56	10252.65	0.63
4	43.98	3.36	580.16	3.32	4358.59	1.49
7	29.57	4.99	340.12	5.66	1444.06	4.50
8	27.87	5.30	299.21	6.43	1202.09	5.41
14	22.95	6.43	175.00	11.00	764.82	8.50
28	15.27	9.67	89.51	21.50	467.13	13.92
56	13.71	10.76	48.07	40.03	272.65	23.84

Table 1: Times and speed-up

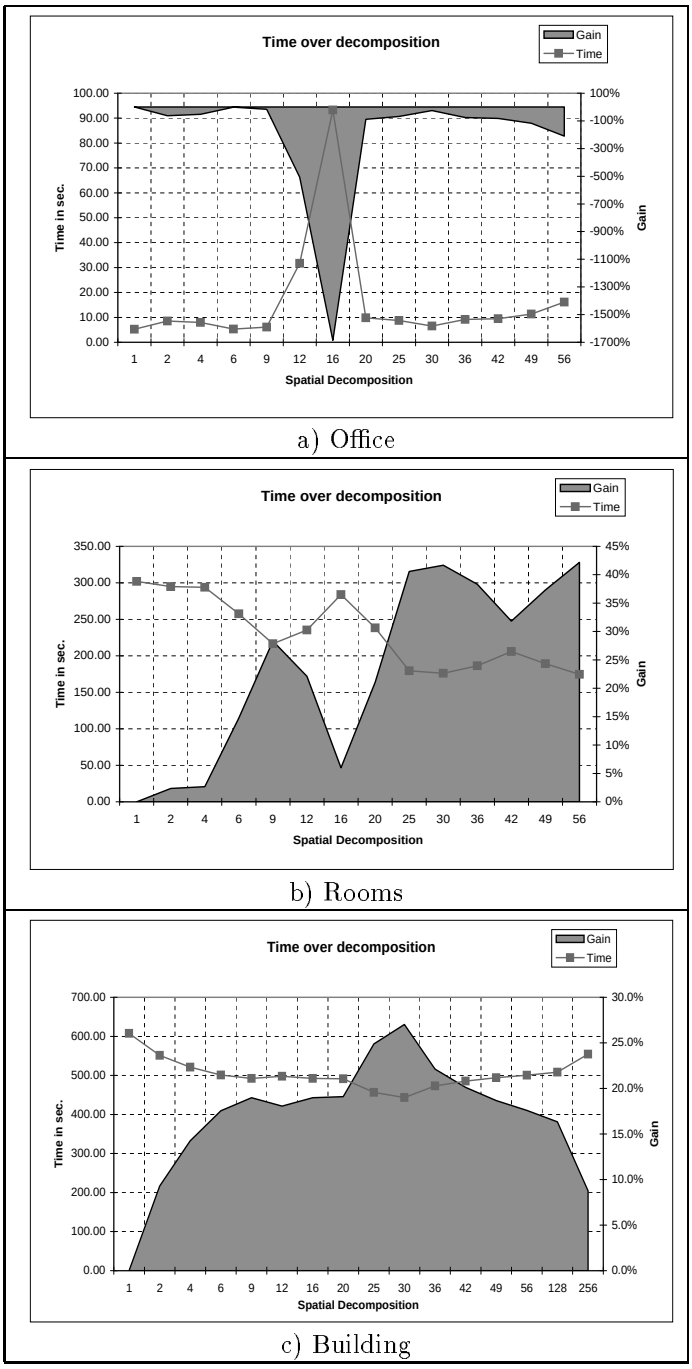


Figure 11: Computing times with a UltraSparc workstation.

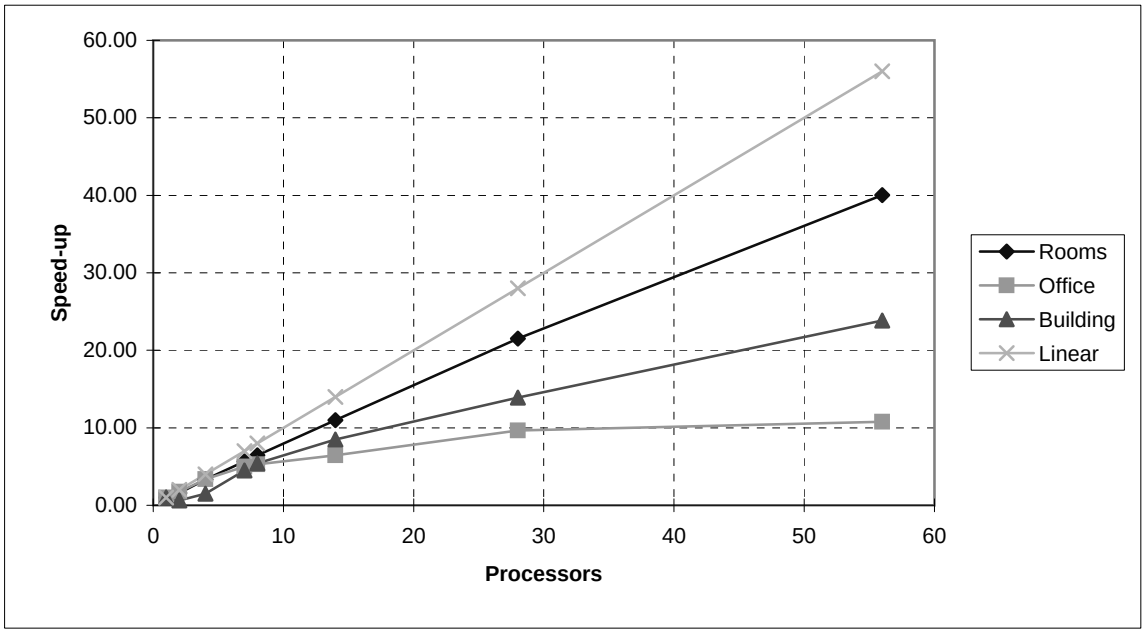


Figure 12: Speed-up

6 Conclusion and perspectives

The techniques we have presented in this paper are being implemented and their development are in an early stage. Several problems are not yet solved such as the mapping of virtual interfaces in order to balance the load among the available processors. Some solutions are being investigated such as load estimation prior to a static data domain decomposition or a dynamic load balancing strategy by moving dynamically virtual interfaces. We plan also to investigate the use of scalable shared memory architecture instead of using message-passing parallel computers. Load balancing issue is easier to solve with shared memory multiprocessors but could be done against data locality. A trade-off will have to be found.

Acknowledgments

We would like to thank the Centre Europeu de Parallelisme de Barcelona (CEPBA) and Prof. Mateo Valero for providing the financial support to this work. X. Pueyo is supported by the project TIC95-0630-C05-05 of the Spanish CICYT.

References

[1] B. Arnaldi, X. Pueyo, and J. Vilaplana. On the division of environments by virtual walls for radiosity computation. In F. Jansen and P. Brunet, editors, *Photorealistic Rendering in Computer Graphics*. Springer Verlag, 1994.

[2] D. Badouel, K. Bouatouch, and T. Priol. Ray tracing on distributed memory parallel computers: Strategies for distributing computation and data. *IEEE Computer Graphics and Application*, 14(4):69–77, July 1994.

[3] D.R. Baum and J.M. Winget. Real time radiosity through parallel processing and hardware acceleration. *Computer Graphics*, 24(2), 1990.

[4] K. Bouatouch, D. Menard, and T. Priol. Parallel radiosity using a shared virtual memory. In *First Bilkent Computer Graphics Conference on Advanced Techniques in Animation Rendering and Visualization*, Ankara, 1993.

[5] E.W. Dijkstra, W.H.J. Feijen, and A.J.M. Van Gasteren. Derivation of a termination detection algorithm for distributed computation. *Inf. Proc. Letters*, 16:217–219, June 1983.

[6] S. Drucker and P. Sckoder. Fast radiosity using a data parallel architecture. In *Proc. of the Third Eurographics Workshop on Rendering*, Bristol, 1992.

[7] M. Feda and W. Purgathofer. Progressive refinement on a transputer array. In F. Jansen and P. Brunet, editors, *Photorealistic Rendering in Computer Graphics*. Springer Verlag, 1994.

[8] A. Fujimoto, T. Tanaka, and K. Iawata. Arts : Accelerated ray tracing system. *IEEE Computer Graphics and Applications*, 6(4):16–26, April 1986.

[9] T. Funkhouser, S. Teller, and D. Khorramabadi. The uc berkeley system for interactive visualization of large architectural models. *Presence, Teleoperators and Virtual environments*, 5(1):13–44, Winter 1996.

[10] G. García and X. Pueyo. Progressive radiosity solutions on simd architectures. Technical Report LSI-94-R, 1994.

[11] S.A Green and D.J. Paddon. Exploiting coherence for multiprocessor ray tracing. *IEEE Computer Graphics and Applications*, 6:12–26, November 1989.

[12] P. Guitton, J. Roman, and C. Schlick. Two parallel approaches for a progressive radiosity. In F. Jansen and P. Brunet, editors, *Photorealistic Rendering in Computer Graphics*. Springer Verlag, 1994.

[13] F.W. Jansen and A. Chalmers. Realism in real-time? In *Proceedings of Fourth Eurographics Workshop on Rendering*. EUROGRAPHICS, 1993. Technical report EG 93 RW.

[14] A. Kok. Grouping of patches in progressive radiosity. In *Proceedings of Fourth Eurographics Workshop on Rendering*. EUROGRAPHICS, 1993. Technical report EG 93 RW.

[15] C. Puech, F. Sillion, and C. Vedel. Improving interaction with radiosity based lighting simulation programs. *Computer Graphics*, 24(2), 1990.

[16] R.J. Recker, D.W. George, and D.P. Greenberg. Acceleration techniques for progressive refinement radiosity. *Computer Graphics*, 24(2), 1990.

[17] E. Reinhard, L.U. Tijssen, and F.W. Jansen. Environment mapping for efficient sampling of the diffuse interreflection. In *Proceedings of Ffth Eurographics Workshop on Rendering*, Darmstadt, 1994.

[18] J. Rohlf and J. Helman. Iris performer: A high performance multiprocessing toolkit for real-time 3d graphics. In *proceedings of SIGGRAPH*, pages 381–394, 94.

[19] H.E. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplifications for indirect illumination calculations. In *Proceedings Graphics Interface'93*, 1993.

[20] F. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Tansactions on Visualization and Computer Graphics*, 1(3), 1995.

[21] F. Sillion and G. Drettakis. Feature-based control of visibility error: A multiresolution clustering algorithm for global illumination. *Computer Graphics Proceedings*, 1995. Annual Conference Series, ACM SIGGRAPH.

[22] B. Smits, J. Arvo, and D.P. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics*, 1994. Annual Conference Series, ACM SIGGRAPH.

[23] W. Sturzlinger, G. Schaufler, and J. Volker. Load balancing for a parallel radiosity algorithm. In *Proc. of ACM Parallel Rendering Symposium'95*, pages 39–45, October 1995.

[24] S. Teller, C. Fowler, and P. Hanrahan. Partitioning and ordering large radiosity computations. *Computer Graphics*, 1994. Annual Conference Series, ACM SIGGRAPH.

[25] R. vanLiere. Divide and conquer radiosity. In F. Jansen and P. Brunet, editors, *Photorealistic Rendering in Computer Graphics*. Springer Verlag, 1994.

[26] A. Varshney and J.F. Prins. An environment projection approach to radiosity for mesh-connected computers. In *Proc. of the Third Eurographics Workshop on Rendering*, Bristol, 1992.

[27] J. Vilaplana. Parallel radiosity solutions based on partial result messages. In *Proc. of the Third Eurographics Workshop on Rendering*, Bristol, 1992.

[28] H. Xu, Q. Peng, and Y. Liang. Accelerated radiosity method for complex environments. In *Proc. of Eurographics'89*, pages 51–61, 1989.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399